AD-A096 546 ANALYTIC SERVICES INC ARLINGTON VA F/G 5/1
RESOURCE ALLOCATION METHODOLOGY FOR AIR FORCE R&D PLANNING.(U)
JUN 80 G COOPER, S ADAMS, J CLARY, J PERLIS F49620-77-C-0025
UNCLASSIFIED ANSER-TDN-80-1 NL

1 OF 2
AD A
096546

AD A096546

2

(2)

TACTICAL DIVISION NOTE

14 ANSER-TDN-80-1

RESOURCE ALLOCATION METHODOLOGY FOR AIR FORCE R&D PLANNING

11 Jun[illegible]80

G. Cooper, Project Leader
S. Adams
J. Clary
J. Perlis

15) [illegible]

Approved by
C. Foreman, Division Manager

anser Analytic Services Inc.
400 Army-Navy Drive, Arlington, Virginia 22202

D

PREFACE

During fiscal 1979, the Director of Program Integration, AF/RDX, tasked ANSER to identify and develop a methodology for allocation of funds among Air Force research, development, and acquisition programs. This report, which consists of four volumes (bound as one document for convenience of user), describes work accomplished in response to that tasking and discusses the resulting resource allocation methodology (RAM). Although RAM has been developed and tested successfully, our experience to date with its use suggests that its future application will be limited unless a method for quantitative evaluation of programs can be devised and implemented.

RESOURCE ALLOCATION METHODOLOGY FOR AIR FORCE R&D PLANNING

Volume 1: Summary Report

CONTENTS
Volume 1

I. INTRODUCTION

This volume is one of four that document ANSER's development of a resource allocation methodology (RAM) for the Director of Program Integration, AF/RDX. Each volume emphasizes a particular aspect of our research and can be read independently of the others. Volume 1 is an overview of the work accomplished. Volume 2 describes the RAM technique itself and how to use the general-purpose computer programs that incorporate it. Volume 3 discusses how to use the interactive computer program developed for use of RAM within AF/RDX. Volume 4 reports on tests made to determine the computational performance of RAM.

In fiscal 1979, AF/RDX requested that ANSER conduct an investigation of analytic techniques that could be used to allocate resources and, if a suitable methodology was found, we were to develop it. The need for such a methodology stems partially from the large number of program elements (PEs) that planners must consider. For example, a single budget could contain over 300 PEs, each of which is proposed at three or four alternative funding levels, called decision packages. Each decision package, which represents some variation on the content of the PE, contributes more or less to the achievement of associated objectives. Objectives include operational needs identified by Mission Area Analysis and other goals that Air Force officials may establish. Furthermore, the Program Objectives Memorandum (POM) is to plan for a 5-year period, which means considering multiple fiscal constraints. Therefore, a large number of factors must be considered.

The basic resource allocation problem can be stated as two questions: At what decision level should each PE be funded to achieve the greatest contributions to the specified objectives while recognizing fiscal constraints over a period of several years? As the availability of fiscal resources increases or decreases, how should the investment strategy be changed to maintain the greatest overall contribution?

We believe we have developed a unique and potentially useful methodology for formulation of investment strategies. However, use of the methodology requires identification of the objectives of interest and evaluation of the contributions of program alternatives to achievement of those objectives. Unless these requirements can be met in a practical way, application of the methodology will be limited. AF/RDX has asked us to address this problem in subsequent research. Chapter II of this volume summarizes our approach to development of the methodology, Chapter III describes how the resource allocation methodology has been and can be used. In Chapter IV, we give our conclusions concerning the utility of the methodology and discuss requirements for its further implementation.

II. THE RESOURCE ALLOCATION METHODOLOGY

A. Our Approach

Our approach to the development of a resource allocation methodology recognizes the multiplicity of objectives that must be considered and their frequently conflicting nature. Furthermore, the benefit from achievement of one objective can rarely be estimated quantitatively in the same terms as those describing other benefits. Therefore, our problem was to develop a methodology that could attempt to simultaneously maximize several incommensurable benefits. This approach contrasts with techniques that attempt to reduce the problem to one in which a single benefit measure is used to develop an investment strategy. One accepted single-benefit technique ranks alternatives by their ratios of benefits and costs; alternatives are then deleted in ascending order from the bottom of the list until the specified budget constraint is no longer exceeded. Unfortunately, the aggregate benefit of the retained alternatives may not be the largest obtainable. Consider, for example, the four alternatives in Table 1, which we ranked in order of benefit/cost ratio. At a sample budget of 18 units, only the first two projects could be funded for a total benefit of 9. However, if the third is substituted for the first, the cost of 17 would be within the allowable budget and the benefit is 10, a number greater than the original allocation.

A second problem exists with use of a single benefit/cost ratio if the cost is obtained by summing multiyear costs, since infeasible cost profiles could easily result. In other words, the sum of the costs of the projects in an R&D program may not violate the sum of the fiscal constraints in various years, but in a given year, those costs could indeed violate the fiscal constraints.

TABLE 1
SAMPLE ALTERNATIVES BY BENEFIT/COST RATIO

Alternative	Cost	Benefit	Benefit/Cost	Cumulative Cost
1	8	6	3/4	8
2	5	3	3/5	13
3	12	7	7/12	25
4	15	5	1/3	40

B. The Allocation Methodology

The resource allocation methodology that we selected after investigation of several techniques employs goal programming. Use of the technique requires definition of a set of goals and their classification into subsets having different priorities for achievement. For example, as a class, operational goals may be afforded a higher priority than that afforded programmatic goals. Furthermore, goals within each subset will usually differ in importance. Consequently, the importance of each goal within every subset must be specified. Specification of the relative priorities of the subsets and the relative importance of the goals in each subset permits use of a priority goal structure for development of an investment strategy. This procedure always finds the investment strategy that most closely satisfies the first-priority goals. The procedure then attempts to satisfy the second-priority goals without degrading the achievement of first-priority goals. This process continues for all successive priority levels. (See Volume 2 for additional explanation of the process.)

This procedure adequately represents all important aspects of the allocation process. It provides substantial flexibility in its representation of the preferences of the decision-maker and in its recognition of the complex, multigoal nature of the decision. Through use of the priority goal structure most practical considerations can be represented. A need to comply with agreements to pursue certain programs can, for example, be represented as a Priority 1 goal. Goals not quantifiable in the same terms can be represented on different priority levels, which precludes any need to compare "apples and oranges". Any type of goal can be incorporated into the structure as long as the user of the method can evaluate the alternatives with respect to it.

We selected an algorithm for determination of the preferred investment strategy that is an adaptation of a technique developed by Petersen.[1] It ranks the alternatives according to an average benefit/cost ratio and then improves the solution by examining various exchanges between the unfunded alternatives and those initially funded. In other words, it finds a workable solution and proceeds systematically to improve it. (See Volume 2 for complete description of the algorithm.)

---

[1] Petersen, Clifford C., "A Capital Budgeting Heuristic Algorithm Using Exchange Operations," *AIIE Transactions*, VI, 2 June 1974.

The technique does not examine all possible combinations of alternatives. Instead, it uses deductive rules to eliminate combinations that need not be considered, realizing substantial efficiency. Observe that the algorithm begins with the type of strategy that results from the techniques discussed in Section II.A and then improves it.

C. Computer Implementation

We wrote two types of computer programs to solve resource allocation problems. The first type consists of general-purpose representations of the methodology, called RAM/GP and RAM/VM. RAM/GP and RAM/VM could be used with any kind of resource allocation problem for which data are available. RAM/GP contains the methodology described in the previous section in which the user wishes to come as close as possible to the specified goals. In some cases, however, the user cannot, or may not want to, specify an explicit objective value as a goal. The user may, instead, simply wish to maximize the contributions made to all goals, which is possible using RAM/VM. This form is equivalent to the previous one in which goals are simply set at unreachable levels. In fact, both programs would find the same solution. However, use of RAM/VM can be more efficient if the less complex data it requires are available. Volume 2 describes both RAM/GP and RAM/VM.

The second type of program is a special-purpose program. It consists of RAM/VM imbedded in a larger, user-interactive program, and was designed to demonstrate the day-to-day use of the method by those responsible for formulating investment strategies. The methodology is tied to the program element (PE) data base used by AF/RDX. By operating a computer terminal, a user obtains access to the data for the programs for which he has responsibility (and only those programs). Then by following simple procedures in response to prompting by the computer, the user can specify alternative budgets for any number of years desired and receive, usually in seconds, the alternatives to be funded that represent the best investment strategy. If he generates alternative strategies in response to revised fiscal constraints, the results are accumulated and displayed in the form of cost/benefit curves. Hard copies of the curves are available as soon as the plots are made. The program is special purpose because it was designed for the type of hardware available at the AF/RDX computer site (Multics computer system) and was tied to the official PE data base as it existed in fiscal 1979. It demonstrates the ease with which the methodology can be used to examine the impacts of enhanced or decremented budget ceilings on the investment strategies. By front loading (shifting funds to earlier periods) or back loading (shifting funds to later periods), the user can examine the impact of the resultant investment strategies in terms of the impact of the progress made toward selected goals. Such excursions make the methodology useful for multiperiod planning on a quick-turnaround basis. We describe this computer program in Volume 3.

We tested the performance of these programs in a process described in Volume 4, examining both the solution accuracy of the programs and the time required to solve problems of various sizes. However, because our methodology is so much more efficient than other available techniques, it was difficult to get a standard for judging the performance of our programs. Testing of this kind is very expensive, and in many cases, the only available commercial routine we could use to establish a solution failed to reach a solution in a practical time. RAM/GP provides practical solution times (on the order of a few minutes) for problem sizes of 200 to 250 decision variables (possible alternatives), while RAM/VM provides practical solution times for larger problems. The practical problem size limitation of the commercial routine was an order of magnitude less and restricted our comparison tests to smaller problems of approximately 45 variables. For problems of this size, statistical analysis indicates a reasonable degree of confidence that RAM results are optimal or near optimal. We believe that the accuracy of results will hold for even larger problems, although we have no statistical basis for that belief.

III. APPLICATIONS OF THE METHODOLOGY

A. Large-Scale Application

We have been unable to apply the methodology on a large-scale primarily because no institutionalized procedures exist within Headquarters USAF for making quantitative evaluations of R&D alternatives. Although at the outset of our work some attempts to formulate and institutionalize such a process had been made,* the process no longer exists.

B. Special-Purpose Applications

RAM has been used for allocating resources in a number of special purpose applications. The methodology has been transferred to computers at the Aeronautical Systems Division (ASD) where it has been used in a pilot study of avionics R&D programs, and where it will be used to allocate resources among aeropropulsion R&D programs.

We have also used RAM within Hq USAF to assist in the development of an armament functional area plan. This functional area cuts across four mission areas: counterair, close air support/battlefield interdiction, defense suppression, and interdiction. We derived goals from appropriate Mission Area Analysis tasks, e.g., targets of various types to be destroyed in various weather conditions. These goals comprised the first priority level and were assigned the relative importance identified in the MAA process. We also identified a single goal at the second-priority level, namely to decrease

*This process is described in "Mission Area Resource Allocation for Air Force R&D," *Defense Systems Management Review*, Volume 2, Spring 1979, procurement spending where possible by competitive bidding. This goal consumes R&D resources to fund multiple contractor involvement in parallel RDT&E efforts.

We formulated funding alternatives that were associated with a unique set of products (armament systems or subsystems) completion dates, risk factors, and cost elements of the decision package sets for the 19 program elements in the functional area.

We determined the impact of each product on the ability of the Air Force to meet the specified operational goals using a linear measure of increase in force effectiveness. Then, this contribution was enhanced or degraded, depending on the timeliness of the product, in meeting the threat and the technical risk involved in developing the product. Summing the contributions of the products of each funding alternative provided a linear index of value for each funding alternative. We then introduced the procurement savings (if any) produced by a funding alternative as a second-priority goal.

We made a number of RAM runs representing excursions on budget ceilings, and benefit/cost curves were produced. Results to date have been well accepted by the tasking office.

IV. CONCLUSIONS

The resource allocation methodology (RAM) that we developed can be used in the formulation of R&D investment strategies (if a practical process for evaluation of alternatives can be devised) and in any type of resource allocation decision that must choose among discrete alternatives. We believe that RAM is a unique technique that can reflect the significant aspects of most allocation decisions. We know of no other technique that formulates and solves the allocation problem in the same manner as that used in RAM.

RAM can be refined and extended in several ways. For example, additional work could produce a general-purpose interactive program of the type demonstrated, but which allows the user to work with more of the data base than the fiscal constraints, e.g., to interactively change goals or costs and evaluations of the alternatives. Also, an output processor should be developed to interpret the changes in results when running excursions on the budget ceilings. Such a processor would highlight the impact on the goals that results when running such excursions. These are essentially data management or bookkeeping improvements, but they would enhance the utility of the method.

Also, while the RAM programs are sufficiently accurate and efficient for the problems we envision, further research could possibly provide some improvements in efficiency if improved efficiency is required. We have found one technique that could improve efficiency, but a major development and test program would be required to verify its potential. An improvement in efficiency could increase the effective problem size of RAM.

Such improvements are of second-order importance; however, the most significant problem in wide-scale use of RAM is the lack of a procedure for evaluating alternatives. If it is feasible to develop and implement such a procedure, we will then be able to provide the decision-maker with an assurance that the best possible strategy with respect to the specified goals has been identified. This strategy will certainly not replace the decision-maker; rather, it will assist him by providing a strategy reflecting those considerations chosen for explicit representation. It will provide the decision-maker with more time to review the resource allocation and, perhaps, to improve it further by applying his own expertise on matters that have not been explicitly assessed.

RESOURCE ALLOCATION METHODOLOGY FOR AIR FORCE R&D PLANNING

Volume 2: Methodology and Computer Programs

CONTENTS
Volume 2

I. INTRODUCTION

This volume is one of four that document ANSER's development of research and development (R&D) resource allocation methodology (RAM) for the Director of Program Integration, AF/RDX. Volume 1 summarizes the work and its applications. Volume 2 describes the RAM technique and how to use the computer programs that incorporate it. Volume 3 describes additional software developed to demonstrate the RAM technique, and Volume 4 describes the way in which we tested the computational capability of the RAM programs. Each volume emphasizes a particular aspect of our research and can be read independently of the others.

This volume concentrates on the technique we adopted to solve the resource allocation problem and the associated computer programs. In Chapter II, we describe what the computer programs do in a mathematical sense, and in Chapter III we describe how to use them. The reader who is not interested in the underlying technical process can probably bypass Chapter II and apply the programs using only the discussion contained in Chapter III. However, some knowledge of computer programming and mathematics is necessary. Chapter IV contains the computer codes for these programs.

In the discussion, we frequently use the vocabulary of the R&D resource allocation problems for which we developed the technique. However, the procedure and computer programs are applicable to a wide range of mathematical optimization problems as described in Chapter II. Although the reader may have to translate our vocabulary to that of his problem, we hope that this volume can be used as a manual for other applications.

II. THE RESOURCE ALLOCATION MODEL

In this chapter, we describe the role of the RAM (Resource Allocation Methodology) computer programs. The discussion is necessarily technical. The reader who is not technically inclined can omit this chapter if he wishes merely to apply the technique, although a scan of at least the first part of Secion II.A, which describes our model of the decision process, is recommended. Section II.B contains a summary of the central RAM algorithm.

A. Mathematical Formulation of the Problem

In the RAM, the decision process involved in allocating resources is modeled as a mathematical optimization problem: the packing problem. This problem can be explained with a physical analogy in three dimensions. Imagine a large box that must hold a number of smaller boxes of varied sizes. Each small box has an associated measure of value, and not all the small boxes will fit simultaneously in the large box. The problem is to pack the large box so that the total value of small boxes contained within is maximized. The three dimensions of this problem correspond to the length, width, and depth of the large box. Note that each small box uses up some of each of these dimensions. In the R&D resource allocation problem, for example, the number of budget periods considered corresponds to the dimensionality of the associated packing problem. Each R&D project alternative (or each small box) potentially uses up some amount in each budget period and some associated measure of benefit.* The problem

*See Volume I for a discussion of benefit measures and other aspects of this problem.

then becomes to maximize the total benefits subject to the budget constraints.

This model of the decision process has three main components. First, there is a set of discrete alternatives that must be selected on a yes or no basis. They are represented by decision variables whose values, typically 0 or 1, correspond to nonselection and selection, respectively. The solution algorithm determines these values. Second, there is an index representing the degree of effectiveness that results when a particular alternative is selected. Finally, there is a measure of the resources that must be consumed in any dimension (such as budget periods) to accrue these benefits. We normally refer to the resource consumption as a "cost", although these resources may not be dollars.

The basic multidimensional packing problem can be further defined as follows:

Let Z represent the total achieved system effectiveness (benefit)

$$\text{Maximize } Z = \sum_{i=1}^{n} A_i X_i$$

Subject to:

$$\sum_{i=1}^{n} C_{ij} X_i \le b_j \quad j=1,\ldots,M$$

$$X_i \in \{0-1\} \ \forall i$$

where:

X_i is a decision variable taking on a value of 1 if the ith option is in the solution, and 0 otherwise.

A_i	is the effectiveness index (a linear measure of benefit) of the ith option.
C_{ij}	is the cost of the ith option with respect to the jth dimensional constraint.
b_j	is the maximum amount of resources available in the jth dimension.

In many cases, this would be a satisfactory representation of a resource allocation decision process. However, for R&D and perhaps many other resource allocation problems, the decision is substantially more complex because it must be made with respect to many objectives or goals. Further, the decision-maker may decide that some goals have a higher priority than others. To reflect these very real complexities, we extended the multidimensional packing problem.

Suppose system performance must be measured against M_r objectives (goals) of which goals 1 to M_1 are in the first priority level, goals $(M_1 + 1)$ to M_2 are in the second priority level, and so on. First-priority goals must be satisfied as far as possible before other goals can be satisfied. Furthermore, goals at the second priority level can be satisfied only if their satisfaction does not degrade the already achieved goals and so on down the line. Each priority level comprises a vector element in the achievement function. Thus we write:

$$\text{Minimize } Z = \left[\sum_{k=1}^{M_1} (W_k N_k), \ldots, \sum_{k=M_{(r-1)}+1}^{M_r} W_k N_k \right]$$

Subject to:

$$\sum_{i=1}^{n} A_{ik} X_i + N_k = G_k \qquad k=1,\ldots,M_r$$

$$\sum_{i=1}^{n} C_{ij} X_i \le b_j \quad j=1,\ldots,M$$

$$X_i \in \{0-1\} \; \forall i$$

where:

X_i	is a decision variable taking on a value of 1 if the ith option is in the solution, and 0 otherwise.
A_{ik}	is the effectiveness index of the ith option relative to the kth goal.
G_k	is the desired effectiveness against the kth goal. $G_k \ge 0$
N_k	are negative deviation variables indicating the amount by which a solution fails to satisfy the kth goal.
W_k	is the weight (relative importance) attached to satisfying the kth goal. These can be viewed as penalty factors. If we fall short of the kth goal by 2 units, the associated element in the achievement vector (to be minimized) is increased by 2 W_k units.
C_{ij}	is the cost of the ith option in the jth dimension (e.g., budget period).

This formulation, called goal programming, is distinguished by the form of the objective function, which minimizes the deviation from specified goals. It also contains a set of weighting factors, $\{W_k\}$, which allow the goals on each priority level to be given a relative importance.

It is not always desirable or possible to specify explicit, quantitative goals. Instead, one may simply wish to find the resource allocation that maximizes the contributions of the chosen alternatives with respect to all the goals. This is equivalent to the previous formulation in which all the goals have been set to a level higher than that obtainable; that is, where:

$$G_k > \sum_{i=1}^{n} A_{ik}X_i \qquad k=1,...,M_r$$

This method is called a vector maximization formulation of the decison process.

Goal programming and vector maximization formulations are fundamentally related, and so their respective programs, RAM/GP (goal programming) and RAM/VM (vector maximization), are very nearly the same. The solution algorithms, the means of determining the X_i, are the same. We simply obtain certain efficiencies in using RAM/VM by recognizing in advance when a maximization model, rather than one in which explicit goals are required, is the best model of the decision process.

One additional refinement of the decison process is incorporated in both RAM/GP and RAM/VM. The subsets that contain mutually exclusive alternatives are incorporated in the solution procedure. In other words, alternatives may be formed into groups, and no more than one can be in the solution. This arrangement models the case in which the decision is to determine, for many programs, which of several predefined alternatives should be selected.

B. Solution Algorithm

The decision models chosen can be adapted to various solutions by a wide range of integer programming techniques. In this class of problems, however, the analyst must choose an acceptable tradeoff between true optimal solutions, problem size, and run-time considerations. Very often, a search for a true optimal solution using one of the classical integer programming techniques results in impractical computer-resource requirements for a problem of any real size. The technique employed in RAM to solve the goal programming and vector maximization packing problems previously defined is based on a variant of the direct-search technique developed by Clifford Petersen at Purdue University.[1] Petersen's method performs to well within the margin of data uncertainty for the R&D resource allocation problem we addressed, while allowing large problems to be solved relatively quickly with very small demands on computer hardware resources. (See Volume 4 for details on the results of a series of test runs made to ascertain RAM performance characteristics relative to certain other techniques.)

The steps for completing the RAM/GP, RAM/VM algorithm are as follows:

1. Compute for each R&D alternative i, its mean proportionate demand on budget funds, R_i, where

$$R_i = \frac{\sum_{j=1}^{m} C_{ij}/b_j}{m}$$

2. Compute the relative independent benefit/cost ratio vector for each alternative, $\overline{AH}_i$, where

$$\overline{AH}_i = \left[\frac{\sum_{K=1}^{M} W_K \, A_{ik}}{R_i}, \ldots, \frac{\sum_{K=M_{(n-1)}+1}^{M_r} W_K \, A_{ik}}{R_i} \right]$$

[1] Petersen, Clifford C., "A Capital Budgeting Heuristic Algorithm Using Exchange Operations," *AIIE Transactions*, VI, 2, June 1974.

3. Pick for each alternative group (subset of mutually exclusive alternatives) the R&D alternative with the highest value $\overline{AH}_i$. Place these alternatives in the "in" set. Rank in descending order of $\overline{AH}_i$ values. Place the remaining alternatives in the "out" set.

4. Delete alternatives from the bottom of the "in" set until budgets are not exceeded in all budget periods. Place deleted alternatives in the "out" set. If problem requires vector maximization formulation go to 5a. If problem requires goal programming formulation go to 5b.

5a. Compute $\overline{A}_i = \overline{AH}_i \cdot R_i$ for all alternatives. Rank the "in" set in order of increasing $\overline{A}_i$ values. Rank the "out" set in order of decreasing $\overline{A}_i$ values.

6a. Subject to dominance rules deriving from the ranking, find the best exchange (2-for-1 or 1-for-1) between the "out" set and the "in" set. If no such exchange improves the solution, go to step 8. (Exchanges must preserve budget feasibility).

7a. Rerank the new "in" and "out" sets as in steps 5a and 6a.

8a. If any funds are left, try to include some level of any unfunded alternative. This is called "Fitback".

9a. STOP

5b. Compute provisional achievement of goal k; g_k for all goals. For each variable in the solution at the Priority 1 level, compute A_{ik}, the contribution of variable i to the satisfaction of goal k.

let $Q_{ik} = \max(0, (a_{ik} + G_k - g_k))$

then let $A_{ik} = \min(a_{ik}, Q_{ik})$

Then compute the total weighted contribution of variable i; $\bar{A}_i$ for the variables in the solution

$$\bar{A}_i = \left[\sum_{k=1}^{M} W_k A_{ik}, \ldots, \sum_{k=M_{(r-1)}+1}^{M_r} W_k A_{ik} \right]$$

Rank the "in" set in order of increasing $\bar{A}_i$ values.

6b. Subject to a weakened set of dominance rules, find the best exchange (2-for-1 or 1-for-1) between the "out" set and the "in" set. To test an exchange for profitability, remove the "in" set variable(s) in its solution, insert the "out" set variable(s) in its place, and an $\bar{A}_i$ value for the new variables as was done in step 5. Compare $\bar{A}_i$s. (Exchanges must maintain budget feasibility.) If no profitable exchange is found, go to step 8b.

7b. Go to step 5b.

8b. If any funds are left, try to include some level of any unfunded alternative (Fitback).

9b. STOP

III. USING THE COMPUTER PROGRAMS

In this section, we describe how to use the RAM/VM and RAM/GP computer programs. These programs solve a resource allocation problem with multiple, prioritized objectives in which constraints are placed on resource use over time. RAM/VM selects the set of alternatives that maximizes the contributions toward the objectives within resource limitations. Most of our experience to date has been with RAM/VM because of its easy application to existing data. Consequently, we devote most of the discussion to this formulation. Because RAM/GP is used in a very similar fashion, we confine our discussion of this formulation to the areas in which it differs from RAM/VM.

These programs are described as being in a free-standing mode, which we used. They are easily imbedded in an interactive program for repetitive use. In fact, we have done this with RAM/VM to demonstrate the process. However, because this demonstration program is useful only on a particular computer and with particular terminals, it is described in Volume 3. RAM/VM and RAM/GP are more general. They have been coded in FORTRAN for use on the Multics computer system, as installed on the Honeywell Series 68/Level 60 computer. Modifications to the coding may be required in the input/output, load/compile, and execute job control language for use on other computing systems.

A. Guide to RAM/VM

1. The Resource Allocation Problem

The RAM/VM program is designed to assist the decision-maker in allocating resources to alternatives in cases where more alternatives exist than can be supported by available resources. Although the problem is quite general, we can describe it in terms of an R&D project selection problem. Each element of the R&D program can be selected at no more than one of a number of predefined, alternative funding levels. Each funding level consumes a different amount of resources (dollars) in each of several budget periods and results in outputs that differ in the benefits received. Benefits reflect the contribution of each alternative to each objective. Benefits measured by the user are used as inputs to this computer program. The objectives of the resource allocation process must be organized into one or more prioritized groups. Within a group, a relative weighting of the objectives may be specified, for example, in order of their relative importance. Finally, resource constraints must be specified for as many budget periods as exist, and the cost of each alternative in each period must be provided.

RAM/VM uses a ranking technique in Phase 1 to obtain an initial feasible solution and then in Phase 2, tries to improve this initial solution through a series of exchange operations.* This approach is based on Petersen's Capital Budgeting Heuristic Algorithm.[2] The program attempts to maximize (with no upper limit) the total achievement value of all objectives at each priority level. The Priority 1 objectives are maximized first, and the objectives associated with each successive priority level (if any exist) are then maximized if the achievement of preceding higher level objectives is not degraded. To maximize the objectives, RAM/VM

[2]Petersen, Clifford C., "A Capital Budgeting Heuristic Algorithm Using Exchange Operations," *AIIE Transactions*, VI, 2, June 1974.

*See Section II.B for description of ranking and exchange techniques.

selects no more than one alternative from each group (mutually exclusive) such that the total measurable benefit (at each priority level) is maximized (subject to the availability of required resources). The alternatives are assigned a value of 1 if included in the solution, 0 if not.

2. Data Input

RAM/VM uses formatted READ statements to input data from a file assigned to logical unit 10. Table 1 identifies the required input to RAM/VM and the corresponding data formats. A brief description of each data item is also included.

Note the flag, IGS. This flag is set equal to 1 if some alternative group exists that must be represented in the solution. For the mutually exclusive alternative groups, which require that no more than one alternative be selected, the IGS set equal to 1 will require that exactly one alternative be selected from each alternative group contained in the Priority 1 objective. This Priority 1 objective must be of the form: Maximize

$$\sum_i A_i X_i ,$$

where A_i is set equal to 1 for alternative i if its associated alternative group must be included in the solution, and A_i is set equal to zero otherwise. Additional comments on the use of IGS are provided in Section III.A.4.

3. Solution Output

The output from RAM/VM is stored in a file assigned to logical unit 8 in a sequential format. "FILEOUT" is the name assigned to the output file by RAM/VM as currently implemented on Multics.

TABLE 1
INPUT DATA FORMATS FOR RAM/VM AND RAM/GP

Card	Data	Format	Description	Comments
1	NVAR	I5	The number of 0-1 variables (total of all alternatives)	
	NG	I5	The number of objectives	
	NP	I5	The number of priority levels	
	NPER	I5	The number of budget periods	
	NGR	I5	The number of mutually exclusive alternative groups	
	IGS	I5	A flag, when set to 1 indicates the presence of alternative groups that must be funded at a nonzero level	
2 – a	W(J)	I5	The weighting factor associated with objective J	Input as (W(J), PR(J), J = 1, NG)
	PR(J)	I5	The priority level that includes objective J	
(a + 1) – b	C(I, L)	10F8.0	The cost of variable I in budget period I	For each budget period I, input as (C(I, I), I =1, NVAR)
(b + 1) – c	IGR(I)	16I5	The alternative group that contains variable I	Input as (IGR(I), I = 1, NVAR)
(c + 1) – d	ILEV(I)	16I5	The funding level within IGR(I) represented by variable I	Input as (ILEV(I), I = 1, NVAR)
(d + 1) – e	B(L)	10F8.0	The budget upper bound for period I	Input as (B(L), I = 1, NPER)
(e + 1) – f	A(I)	10F8.0	The contribution of variable I to the achievement of an objective	For each objective, input as (A(I), I = 1, NVAR)
*(f + 1) – g	RHS(J)	10F8.0	The desired achievement value of each goal or objective	Input as (RHS(J), J = 1, NG)

*RHS(J) is input for RAM/GP only.

"FILEOUT" provides the initial solution obtained by RAM/VM under Phase 1 and the final solution obtained under Phase 2. The results shown are the variables contained in the solution and the corresponding alternative group number (mutually exclusive group) and funding alternative number (alternative within the associated group), the achievement vector for each priority level, and the slack (or unused resources) remaining in each period.

4. Sample Problem

Table 2 is a sample data file for RAM/VM, which is set up in the format identified in Table 1. Line (1) sets the parameters for the problem. There are nine 0-1 variables (total number of alternative groups) (NVAR=9), two objectives (NG=2), one priority level (NP=1), three time periods (NPER=3) and three alternative groups (NGR=3). The flag, IGS, is set to zero (IGS=0), indicating that for this problem no alternative groups need be represented in the solution.

TABLE 2
SAMPLE DATA FILE FOR RAM/VM (IGS = 0)

```
(1)     9       2       1       3       3       0
(2)     1       1       2       1
(3)  32.000   2.000  30.000  10.000  98.000   0.000  38.000  89.000  71.000
(4)  96.000  25.000  33.000  35.000  72.000  48.000  94.000  92.000  96.000
(5)   7.000  92.000  31.000  84.000   0.000  90.000  26.000  11.000  47.000
(6)     1       1       2       2       2       2       3       3       3
(7)     1       2       1       2       3       4       1       2       3
(8) 200.000 190.000 200.000
(9)  69.000  13.000  32.000   2.000  30.000  10.000  98.000   0.000  38.000
(10  89.000  71.000  96.000  25.000  33.000  35.000  72.000  48.000  94.000
```

Line (2) shows that each objective is at the first priority level (PR(J)=1), J=1,2); however, the second objective is shown to be twice as important as the first. Therefore, the weighting factors are 1 for the first objective (W(1)=1) and 2 for the second objective (W(2)=2).

Lines (3) through (5) identify the quantity of resources required by each alternative during each time period.

Lines (6) and (7) identify which variables are associated with each alternative group. In this example, there are three such groups, the first group having two alternatives, the second group having four alternatives, and the third group having three alternatives.

Line (8) identifies the total resources available during each of the three time periods.

Lines (9) and (10) identify the contribution of each alternative to the achievement of the two objectives.

Mathematically, the sample problem is set up as follows:

Maximize

$$Z = 1\ (69X_1 + 13X_2 + 32X_3 + 2X_4 + 30X_5 + 10X_6 + 98X_7 + 0X_8 + 38X_9)$$
$$+\ 2\ (89X_1 + 71X_2 + 96X_3 + 25X_4 + 33X_5 + 35X_6 + 72X_7 + 48X_8 + 94X_9)$$

(where Z is the objective function, and the coefficients represent the contribution of each variable (alternative) to the achievement of each objective.)

Subject to the following resource constraints:

$$32X_1 + 2X_2 + 30X_3 + 10X_4 + 98X_5 + 0X_6 + 38X_7 + 89X_8 + 71X_9 \leq 200.0$$
$$96X_1 + 25X_2 + 33X_3 + 35X_4 + 72X_5 + 48X_6 + 94X_7 + 92X_8 + 96X_9 \leq 190.0$$
$$7X_1 + 92X_2 + 31X_3 + 84X_4 + 0X_5 + 90X_6 + 26X_7 + 11X_8 + 47X_9 \leq 200.0$$

(where each equation represents one time period, the coefficients identify the resources required by each variable (alternative), and the righthand side provides the maximum quantity of the resources available in each time period.)

The constraint:

$$X_1 = \{0,1\} \quad i=1,9$$

requires that an alternative either be selected or rejected. The constraints:

$$X_1 + X_2 \le 1 \quad \text{(Alternative Group 1)}$$

$$X_3 + X_4 + X_5 + X_6 \le 1 \quad \text{(Alternative Group 2)}$$

$$X_7 + X_8 + X_9 \le 1 \quad \text{(Alternative Group 3)}$$

require no more than one alternative to be selected from each alternative group. It is possible that no alternatives will be selected from a given group.

The solution to this problem is shown in Table 3. Note the improvement in the achievement vector in Phase 2 over that in Phase 1. RAM/VM found an initial feasible solution and then, through a two-for-one exchange, improved the value of the objective function. The exchange replaced variable index 1 with variable indices 2 and 7. The value of the achievement vector in the final solution is 621.0. This is the value of Z with X_2, X_3, and X_7 (the alternatives associated with the variable indices 2, 3, and 7) set equal to 1 and all other variables set to 0; this value represents the optimal (or near optimal) achievement with the set of constraints. These variable indices (2, 3, and 7) represent Alternative Group 1 (Alternative 2), Alternative Group 2

TABLE 3
SAMPLE OUTPUT FROM RAM/VM

```
                    multiple objective resource allocation solution system output
                                      output from phase 1

                          the following variables are in the solution

variable index               alternative group number                funding alternative number
      3                                 2                                        1
      1                                 1                                        1

achievement vector
   priority 1                 471.0000
                    multiple objective resource allocation solution system output
                                      output from phase 2

                          the following variables are in the solution

variable index               alternative group number                funding alternative number
      2                                 1                                        2
      3                                 2                                        1
      7                                 3                                        1

achievement vector
   priority 1                 621.0000
                    multiple objective resource allocation solution system output

                     slack remaining in period  1:              130.000000

                     slack remaining in period  2:               38.000000

                     slack remaining in period  3:               51.000000
```

(Alternative 1) and Alternative Group 3 (Alternative 1). With problems of more than one priority level, the achievement vector will be provided for each associated priority level. The output also identifies the slack, or quantity of unused resources, remaining in each time period.

If, as an additional constraint, one of the alternatives in the third alternative group had to be selected, IGS would be set equal to 1, and an additional objective would be added. This objective would be at the Priority 1 level and would be:

$$\text{Maximize:}\quad 1X_7 + 1X_8 + 1X_9 .$$

The other objectives would then be treated as Priority 2 objectives, and the objective function Z would be:

Maximize

$$\begin{aligned} Z = \Big[&(0X_1 + 0X_2 + 0X_3 + 0X_4 + 0X_5 + 0X_6 + 1X_7 + 1X_8 + 1X_9);\\ &\big(1(69X_1 + 13X_2 + 32X_3 + 2X_4 + 30X_5 + 10X_6 + 98X_7 + 0X_8 + 38X_9)\\ &+ 2(89X_1 + 71X_2 + 96X_3 + 25X_4 + 33X_5 + 35X_6 + 72X_7 + 48X_8 + 94X_9)\big)\Big] \end{aligned}$$

Table 4 shows how the data file would be set up for such a problem. Table 4 represents the same problems as those in Table 2, except it includes the requirement to select one of the alternatives from the third alternative group.

5. Limitations Due to Size of Problem

The array dimensions restrict the problem size to 200 goals, 3 priority levels, 10 budget periods, 475 alternatives, and 150 alternative groups. The problem size can be increased through appropriate redimensioning of the arrays, subject only to time and system limitations.

TABLE 4
SAMPLE DATA FILE FOR RAM/VM (IGS = 1)

```
    9       2       2       3       3       1
    1       1       1       2       2       2
  32.000    2.000  30.000  10.000  98.000   0.000  38.000  89.000  71.000
  96.000   25.000  33.000  35.000  72.000  48.000  94.000  92.000  96.000
   7.000   92.000  31.000  34.000   0.000  90.000  26.000  11.000  47.000
    1       1       2       2       2       2       3       3       3
    1       2       1       2       3       4       1       2       3
 200.000  190.000 200.000
   0.000    0.000   0.000   0.000   0.000   0.000   1.000   1.000   1.000
  69.000   13.000  32.000   2.000  30.000  10.000  98.000   0.000  38.000
  89.000   71.000  96.000  25.000  33.000  35.000  72.000  48.000  94.000
```

6. Computational Experience

We ran a number of randomly generated problems on RAM/VM during a test program. All the problems generated had four goals, one priority level, and four budget periods. Table 5 provides a summary of some of the test program results.*

B. Guide to RAM/GP

RAM/GP is a computer code similar to RAM/VM in that it also solves the multidimensional, multiple-objective packing problem and uses the same ranking technique and exchange heuristic. RAM/GP, however, embodies more aspects of the goal programming methodology for solving the multiple objective problem. RAM/VM maximizes the achievement value of all objectives for each priority level; whereas RAM/GP minimizes the deviation from set objectives or goals for each priority level.

*See Volume 4 for additional test data.

TABLE 5
SUMMARY OF TEST PROGRAM RESULTS

Number of Variables	Mean Percentage of Optimum*	Mean CPU Time (CPU secs)
21	97.3	1.44
40	97.7	2.21
60	–†	3.75
75	–†	6.71
99	–†	9.10

*We based percentage of optimum on RAM/VM solutions that were within a percentage of an upper bound on the optimal solution. The RAM/VM solution may, therefore, be closer to the true optimal.

†Limitations of the test program restricted the determination of the upper bound on the optimal solution for problems with more than 50 variables.

Data input for RAM/GP differs from that for RAM/VM only in that the desired achievement value for each objective is included in the data input. This value is shown in Table 1 as RHS(J), the last line of data to be input. If the objectives (RHS(J)) are set high enough (so that they cannot be attained), RAM/GP will arrive at the same solution as that of RAM/VM. RAM/GP does not give any credit for surpassing goals; therefore, the value of the achievement vector as provided by the output will never exceed the sum of all goals,

$$\left(\sum_{J=1}^{NG} \text{RHS}(J)\right),$$

even though the set goal for some (or all) objectives may be exceeded.

The information provided by the output from RAM/GP is the same as that provided by RAM/VM output. The solutions obtained from RAM/GP during the test program compared favorably with those of RAM/VM and should be within 2 to 3 percent of optimum for problems with less than 50 variables. RAM/GP does require more CPU time to arrive at the solution than does RAM/VM.

IV. COMPUTER CODES

This chapter contains the computer program listings for RAM/GP and RAM/VM.

A. RAM/GP

```
C************************ PROGRAM RAM/GP *****************************
C
C ****SUBROUTINE FUNCTIONS
C
C         1  FEAS--CHECKS AN EXCHANGE OPERATION FOR BUDGET FEASIBILITY.
C
C         2  COMPARE--CHECKS AN OPERATION FOR PROFITABILITY.
C
C         3. RERANK--REORDERS THE PROJECTS IN THE SETS JS AND NS AFTER
C            EACH EXCHANGE. (JS IS RANKED IN ASCENDING ORDER OF PROFIT,
C            NS IN DESCENDING ORDER)
C
C         4. FITBACK--USES UP BUDGET SLACK BY ADDING ANY FEASIBLE PROJECTS
C            (REGARDLESS OF RELATIVE PROFIT) TO JS.
C
C********************************************************************
C
C ****INPUT VARIABLES
C
C         1. NVAR--THE NUMBER OF 0-1 VARIABLES.
C         2. NG--THE NUMBER OF OBJECTIVES.
C         3. NP--THE NUMBER OF PRIORITY LEVELS.
C         4. NPER--THE NUMBER OF BUDGET PERIODS.
C         5. NGR--THE NUMBER OF MUTUALLY EXCLUSIVE GROUPS (ALTERNATIVE GROUPS).
C         6. W(J)--THE WEIGHTING FACTOR ASSOCIATED WITH OBJECTIVE J.
C         7. PR(J)--THE PRIORITY LEVEL WHICH INCLUDES OBJECTIVE J.
C         8. IGR(I)--THE ALTERNATIVE GROUP WHICH CONTAINS VARIABLE I.
C         9. A(I)--THE CONTRIBUTION OF VARIABLE I TO THE ACHIEVEMENT OF
C            AN OBJECTIVE.
C        10. C(I,L)--THE COST OF VARIABLE I IN BUDGET PERIOD L.
C        11. B(L)--THE BUDGET UPPER BOUND FOR PERIOD L.
C        12. ILEV(I)--THE FUNDING LEVEL WITHIN IGR(I) REPRESENTED BY VARIABLE I.
C        13. IGS--IF SET TO 1, INDICATES THE PRESENCE OF ALTERNATIVE GROUPS
C             WHICH MUST BE FUNDED AT SOME NON-ZERO LEVEL.
C
C**********************************************************************
```

```
C****INTERNAL VARIABLES
C
C        1. R(I)--THE PROPORTION OF AVAILABLE FUNDS REQUIRED BY
C           VARIABLE I,AVERAGED OVER ALL BUDGET PERIODS.
C        2. AH(I,K)--FOR EACH VARIABLE I,THIS IS AN NP-DIMENSIONAL VECTOR,
C           WHOSE KTH COMPONENT IS A WEIGHTED SUM OF CONTRIBUTIONS OF VARIABLE I
C           TO THE OBJECTIVES AT PRIORITY LEVEL K.
C        3. INIF--A FLAG WHICH COMMUNICATES THE CURRENT STAGE TO THE
C           SUBROUTINE 'COMPARE'.
C        4. INC(I)--HAS A VALUE OF 1 IF VARIABLE I IS IN THE SOLUTION;
C           O OTHERWISE.
C        5. IGR(M)--HAS A VALUE OF 1 IF THE GROUP M IS REPRESENTED IN
C           THE SOLUTION;O OTHERWISE.
C        6. JS(M)--TAKES ON THE INDEX NUMBER OF THE MTH VARIABLE
C           IN THE SOLUTION
C        7. JL--THE NUMBER OF VARIABLES CURRENTLY IN THE SOLUTION SET;JS(M).
C        8. NS(M)--TAKES ON THE INDEX NUMBER OF THE MTH VARIABLE NOT IN THE
C           SOLUTION.
C        9. LL--THE NUMBER OF VARIABLES CURRENTLY OUT OF THE SOLUTION.
C       10  SLACK(L)--A VECTOR WHICH INDICATES UNSPENT FUNDS IN
C           EACH BUDGET PERIOD FOR THE CURRENT SOLUTION.
C       11. NUMF--IS A FLAG WHICH INDICATES THE TYPE OF EXCHANGE
C           TO PERFORM.
C       12. PROF(K)--IS A NP-DIMENSIONAL VECTOR WHOSE KTH COMPONENT
C           IS THE PROFIT GAINED IN THE KTH PRIORITY LEVEL SO FAR IN
C           THE CURRENT EXCHANGE CYCLE.
C       13. PROFN(K)--IS A NP-DIMENSIONAL VECTOR WHOSE KTH COMPONENT IS THE
C           PROFIT IN THE KTH PRIORITY LEVEL FOR THE EXCHANGE
C           UNDER CONSIDERATION.
C       14. IFLAG--TAKES ON A VALUE OF 1 IF AN EXCHANGE IS PROFITABLE;
C           O OTHERWISE.
C       15. JFLAG--TAKES ON A VALUE OF 1 IF AN EXCHANGE IS FEASIBLE; O OTHERWISE
C       16. CAND(I)--TAKES ON THE INDEX NUMBER OF THE ITH VARIABLE
C           IN THE FITBACK SET.
C
C***********************************************************************

C****CURRENT DIMENSION STATUS
C
C           1. NUMBER OF GOALS: 200
C
C           2. NUMBER OF PRIORITY LEVELS: 3
C
C           3. NUMBER OF BUDGET PERIODS : 10
C
C           4. NUMBER OF 0-1 VARIABLES(ALTERNATIVES): 475
C
C           5. NUMBER OF ALTERNATIVE GROUPS: 150
C
C   **********************************************************************
C
C              RAM/GP WAS DEVELOPED AND WRITTEN AT
C
C                   ANALYTIC SERVICES, INC
C                    (TACTICAL DIVISION)
C                    400 ARMY-NAVY DRIVE
C                 ARLINGTON, VIRGINIA  22202
C
C                      (703) 979-0700
C                       AV   225-5640
C
C   **********************************************************************
```

```
      DIMENSION R(475)
      INTEGER PR(475),W(475)
      INTEGER OV,HV,OV1,OV2
      COMMON /COMM1/ SL(10),C(200,10),NPER,JFLAG,SLACK(10)
      COMMON /COMM2/ JS(50),JL
      COMMON /COMM3/ AH(200,3),PROF(3),IFLAG,NP,PROFN(3)
      COMMON /COMM4/ IGN(50),INC(200),CAND(50),NGR,NVAR
     &,LK,ILEV(200),ACH(3)
      COMMON /COMM5/ NUMF,OV,JV,KV,LV,MV,HV,IOUT1,IOUT2
     & ,IOUT3,IN1,IN2,IN3,NL,NM,NH,II,JJ,KK
     &,IA,IB,IC,IAA,IBB,ICC
      COMMON /COMM6/ NS(200),LL,IGR(200)
      COMMON /COMM7/ INIF
      COMMON /COMM8/ B(10)
      COMMON /COMM9/ A(200,50),RHS(50),PER(50),EXT(50),JW(200),
     &NG,IPR(200)
      READ(10,501) NVAR,NG,NP,NPER,NGR
      READ(10,502) (JW(J),IPR(J),J =1,NG)
      DO 737 L=1,NPER
 737  READ(10,503) (C(I,L),I=1,NVAR)
      READ(10,502) (IGR(I),I=1,NVAR)
      READ(10,502) (ILEV(I),I = 1,NVAR)
      READ(10,503) (B(L),L=1,NPER)
      DO 714 J=1,NG
 714  READ(10,503) (A(I,J),I=1,NVAR)
      READ(10,503) (RHS(J),J=1,NG)
      ISIZ=1
 501  FORMAT(16I5)
 502  FORMAT(16I5)
 503  FORMAT(10F8.3)
 996  FORMAT(' ',25I5)
C   1. CALCULATE R(I)
      NVAR=NVAR+1
      NGR=NGR+1
      DO 904 K=1,NG
      A(NVAR,K) =0.
 904  CONTINUE
      DO 905 K=1,NP
      AH(NVAR,K)=0.
 905  CONTINUE
      IGR(NVAR)=NGR
      ILEV(NVAR)=1
      DO 906 L=1,NPER
      C(NVAR,L)=999999.
 906  CONTINUE
      DO 1  I = 1,NVAR
      R(I)=0.0
      DO 2  L = 1,NPER
      R(I)=R(I)+(C(I,L)/B(L))
 2    CONTINUE
      R(I) = R(I)/NPER
 1    CONTINUE
C  2. CALCULATE AH(I,K)
      DO 4 K=1,NP
      DO 4   I = 1,NVAR
      AH(I,K) = 0.0
 4    CONTINUE
      DO 3  J = 1,NG
      K=IPR(J)
      DO 5  I = 1,NVAR
      AA = (A(I,J)*JW(J))/R(I)
      AH(I,K) = AH(I,K) + AA
 5    CONTINUE
 3    CONTINUE
C   DETERMINE INITIAL SOLUTION
```

```
C   1. GET MAX(A/R)I FOR EACH GROUP.
      INIF = 1
      LL = 0
      DO 6  M = 1,NGR
      MAX=0
      DO 7  I=1,NVAR
      IF(IGR(I) .NE. M) GO TO 7
      INC(I)=0
      IV=I
      IF(MAX.EQ.0) GO TO 101
      CALL COMPARE (IV,OV,INIF)
      IF(INC(I).EQ.0) GO TO 102
      LL=LL+1
      NS(LL)=OV
      OV = I
      GO TO 7
 102  LL = LL + 1
      NS(LL) = I
      GO TO 7
 101  OV=I
      INC(I)=1
      MAX=1
 7    CONTINUE
      JS(M)=OV
 6    CONTINUE
      INIF=0
C*******RANK (JS)*****************
      DO 8  M=2,NGR
      N=M
 103  MO=JS(N-1)
      MN=JS(N)
      CALL COMPARE(MN,MO,INIF)
      IF(IFLAG .EQ. 0) GO TO 8
      INTER = JS(N-1)
      JS(N-1)=JS(N)
      JS(N)=INTER
      N=N-1
      IF(N-1) 8,8,103
 8    CONTINUE
      JL=NGR
C***CHECK (JS) FOR FEASIBILITY AND DELETE BOTTOM ENTRY IF NOT FEASIBLE
 104  DO 9 L = 1,NPER
      CC=0.0
      DO 10 M = 1,JL
      I=JS(M)
      CC=CC+C(I,L)
      IF(CC.GT.B(L)) GO TO 105
 10   CONTINUE
      SLACK(L)=B(L)-CC
 9    CONTINUE
      GO TO 106
 105  II = JS(JL)
      INC(II)=0
      JL=JL-1
      LL=LL+1
      NS(LL)=II
      GO TO 104
 106  DO 11 K = 1,NP
      DO 11 I = 1,NVAR
      AH(I,K)=AH(I,K)*R(I)
 11   CONTINUE
      DO 769 I=1,LL
      J=NS(I)
 769  CONTINUE
      CALL RERANK
      DO 797 I=1,LL
      J=NS(I)
 797  CONTINUE
 500  CONTINUE
      KKFL=0
      CALL FITBACK(KKFL)
      IOFL=1
      CALL OUTPUT(IOFL)
C*******SECTION 2    FIRST EXCHANGE*********
      DO 722 I=1,LL
      J=NS(I)
 722  CONTINUE
 116  CONTINUE
      DO 12 K = 1,NP
      PROF(K)=0.0
 12   CONTINUE
      NV2=NVAR+2
      INIF=3
      DO 13 NL=1,LL
      NH=NL+1
      JJFL=1
 109  IF(NH.GT.LL) GO TO 13
      HV = NS(NH)
      LV=NS(NL)
      IF(IGR(HV) .EQ. IGR(LV)) GO TO 107
      GO TO 108
 107  NH=NH+1
      GO TO 109
 108  CONTINUE
      DO 15 L=1,NPER
      C(NV2,L)=C(HV,L)+C(LV,L)
 15   CONTINUE
      DO 16 II=1,JL
      OV=JS(II)
      IS=IGR(LV)
      IX=IGR(HV)
      IF(IGN(IS).NE.0.AND.IGR(LV).NE.IGR(OV)) GO TO 16
      DO 30 K=1,NP
      AH(NV2,K)=0.
      AH(LV,K)=0.
 30   CONTINUE
      DO 14 J=1,NG
      R1=A(LV,J)+A(HV,J)
      D1=A(LV,J)
      RNEED=EXT(J)+A(OV,J)
      IF(RNEED.LT.0.)RNEED=0.
      R2=AMIN1(R1,RNEED)
      D2=AMIN1(D1,RNEED)
      LY=IPR(J)
      AH(NV2,LY)=AH(NV2,LY)+(R2*JW(J))
      AH(LV,LY)=AH(LV,LY)+(D2*JW(J))
 14   CONTINUE
      IF(IGN(IX).NE.0.AND.IGR(HV).NE.IGR(OV)) GO TO 110
      NV=NV2
      CALL COMPARE(NV,OV,INIF)
      IF(IFLAG.EQ.0) GO TO 111
      CALL FEAS(OV,NV)
      IF(JFLAG.EQ.0) GO TO 110
      NUMF =2
      GO TO 112
```

```
110   IF(JJFL.GT.1) GO TO 16
      NV=LV
      IF(NV.EQ.LNM.AND.OV.EQ.LNN) GO TO 16
      CALL COMPARE(NV,OV,INIF)
      LNM=NV
      LNN=OV
      IF(IFLAG.EQ.0) GO TO 16
      CALL FEAS(OV,NV)
      IF(JFLAG .EQ. 0) GO TO 16
      NUMF=1
      GO TO 112
111   NH=NH+1
      JJFL=2
      GO TO 109
112   DO 17 K = 1,NP
      PROF(K)=PROFN(K)
17    CONTINUE
      IA = II
      IOUT=OV
      IN1 = LV
      IAA = NL
      IF(NUMF.NE.2) GO TO 113
      IN2 = HV
      IBB = NH
      GO TO 111
113   IN2 = 0
16    CONTINUE
      GO TO 107
13    CONTINUE
      DO 18 K=1,NP
      IF (PROF(K).GT.0.0) GO TO 114
18    CONTINUE
      KKFL = 1
      CALL FITBACK(KKFL)
      IOFL=2
      CALL OUTPUT(IOFL)
      CALL RERANK
      GO TO 118
114   INC(IOUT)=0
      INC(IN1)=1
      INTER3=JS(IA)
      JS(IA)=NS(IAA)
      N=JS(IA)
      IJ=IGR(N)
      NS(IAA)=INTER3
      IP=IGR(INTER3)
      IGN(IP)=0
      IGN(IJ)=1
      IF(IN2.EQ.0) GO TO 115
      LLL=0
      LL=LL-1
      DO 396 I=1,LL
      J=I+1
      IF(NS(I).EQ.IN2) LLL=1
      IF(LLL.EQ.1) NS(I)=NS(J)
396   CONTINUE
      INC(IN2)=1
      JL=JL+1
      JS(JL)=IN2
      LX=IGR(IN2)
      IGN(LX)=1
```

```
 115   CALL RERANK
       DO 28 L=1,NPER
       CC=0.0
       DO 29 M=1,JL
       I=JS(M)
       CC=CC+C(I,L)
 29    CONTINUE
       SLACK(L)=B(L)-CC
 28    CONTINUE
       GO TO 116
 118   CONTINUE
 810   IOFL=3
       CALL OUTPUT(IOFL)
       CLOSE(8)
       CLOSE(10)
       STOP
       END
C*****SUBROUTINES*****************************
       SUBROUTINE FEAS(LOV,NV)
       COMMON /COMM1/ SL(10),C(200,10),NPER,JFLAG,SLACK(10)
       COMMON /COMM7/ INIF
       DO 1 L=1,NPER
       SL(L) = SLACK(L)
       SL(L)=SL(L)+C(LOV,L)-C(NV,L)
       SLF=SL(L)+.0005
       IF(SLF) 10,1,1
 1     CONTINUE
       JFLAG=1
       GO TO 11
 10    JFLAG=0
 11    RETURN
       END
C*******RERANK (JS) AND (NS)***************
       SUBROUTINE RERANK
       COMMON /COMM2/ JS(50),JL
       COMMON /COMM7/ INIF
       COMMON /COMM3/ AH(200,3),PROF(3),IFLAG,NP,PROFN(3)
       COMMON /COMM4/ IGN(50),INC(200),CAND(50),NGR,NVAR
      &,LK,ILEV(200),ACH(3)
       COMMON /COMM6/ NS(200),LL,IGR(200)
       COMMON /COMM9/ A(200,50),RHS(50),PER(50),EXT(50),JW(200),
      &NG,IPR(200)
       INIF = 0
       DO 1 J=1,NG
       PER(J)=0.
 1     CONTINUE
       DO 2 IQ=1,JL
       DO 2 J=1,NG
       I=JS(IQ)
       PER(J)=PER(J)+A(I,J)
 2     CONTINUE
       DO 3 J=1,NG
       EXT(J)=RHS(J)-PER(J)
 3     CONTINUE
       DO 4 I=1,JL
       DO 4 K=1,NP
       LZ=JS(I)
       AH(LZ,K)=0.
 4     CONTINUE
       QL=0.
       DO 5 IQ=1,JL
       DO 5 J=1,NG
       I=JS(IQ)
      RV=A(I,J)+EXT(J)
      IF(RV.LT.O.) RV=0.
      APS=A(I,J)
      RSV=AMIN1(APS,RV)
      K=IPR(J)
      AH(I,K)=AH(I,K)+RSV*JW(J)
 5    CONTINUE
      IF(JL.LT.2) GO TO 71
      DO 70 M=2,JL
      N=M
 700  MO=JS(N-1)
      MN=JS(N)
      CALL COMPARE(MN,MO,INIF)
      IF(IFLAG.EQ.1) GO TO 70
      INTER=JS(N-1)
      JS(N-1)=JS(N)
      JS(N)=INTER
      N=N-1
      IF(N-1) 70,70,700
 70   CONTINUE
 71   CONTINUE
C************************(NS)***************
      DO 57 M = 2,LL
      N=M
 604  MO=NS(N-1)
      MN=NS(N)
      CALL COMPARE(MN,MO,INIF)
      IF(IFLAG.EQ.0) GO TO 57
      INTER=NS(N-1)
      NS(N-1) = NS(N)
      NS(N) = INTER
      N=N-1
      IF(N-1) 57,57,604
 57   CONTINUE
      DO 100 I=1,JL
      II=JS(I)
 100  CONTINUE
      RETURN
      END
C***********EXECUTE FITBACK*****************
      SUBROUTINE FITBACK (KKFL)
      COMMON /COMM1/ SL(10),C(200,10),NPER,JFLAG,SLACK(10)
      COMMON /COMM2/ JS(50),JL
      COMMON /COMM4/ IGN(50),INC(200),CAND(50),NGR,NVAR
     &,LK,ILEV(200),ACH(3)
      COMMON /COMM6/ NS(200),LL,IGR(200)
      COMMON /COMM7/ INIF
      DO 15 L=1,NPER
      SL(L)=SLACK(L)
 15   CONTINUE
      DO 11 M=1,NGR
      IGN(M)=0
 11   CONTINUE
      DO 10 I=1,JL
      N=JS(I)
      M=IGR(N)
      IGN(M)=1
 10   CONTINUE
      LK=0
      DO 12 I=1,LL
      II=NS(I)
      M=IGR(II)
      IF(IGN(M).NE.0) GO TO 12
      DO 13 L=1,NPER
      IF(C(II,L).GT.SL(L)) GO TO 12
13    CONTINUE
      LK=LK+1
      CAND(LK)=II
      DO 14 L=1,NPER
      SL(L)=SL(L)-C(II,L)
14    CONTINUE
12    CONTINUE
      IF(KKFL .EQ. 0 .OR. LK .EQ. 0) GO TO 17
      DO 18 I = 1,LK
      JJ = CAND(I)
      INC(JJ) =1
      JL = JL + 1
      LL = LL-1
      JS(JL) = JJ
      JK = IGR(JJ)
      IGN(JK) =1
18    CONTINUE
17    CONTINUE
      RETURN
      END
      SUBROUTINE COMPARE(IN,IO,INIF)
      COMMON /COMM3/ AH(200,3),PROF(3),IFLAG,NP,PROFN(3)
      COMMON /COMM4/ IGN(50),INC(200),CAND(50),NGR,NVAR
     &,LK,ILEV(200),ACH(3)
      COMMON /COMM2/ JS(50),JL
      IF(INIF .NE. 1) GO TO 11
      DO 1 K = 1,NP
      IF(AH(IN,K) - AH(IO,K)) 2,1,3
1     CONTINUE
2     GO TO 10
3     INC(IN) = 1
      INC(IO) = 0
      GO TO 10
11    IF(INIF .NE. 0) GO TO 12
      DO 4 K = 1,NP
      IF(AH(IN,K) - AH(IO,K)) 5,4,6
4     CONTINUE
5     IFLAG = 0
      GO TO 10
6     IFLAG = 1
      GO TO 10
12    IF(INIF .NE. 3) GO TO 10
      DO 7 K = 1,NP
      PROFN(K) = AH(IN,K)-AH(IO,K)
7     CONTINUE
      DO 8 K = 1,NP
      IF(PROFN(K)-PROF(K)) 18,8,20
8     CONTINUE
18    IFLAG = 0
      GO TO 10
20    IFLAG = 1
10    RETURN
      END
      SUBROUTINE OUTPUT(IOFL)
      COMMON /COMM4/ IGN(50),INC(200),CAND(50),NGR,NVAR
     &,LK,ILEV(200),ACH(3)
```

```
      COMMON /COMM2/ JS(50),JL
      COMMON /COMM3/ AH(200,3),PROF(3),IFLAG,NP,PROFN(3)
      COMMON /COMM6/ NS(200),LL,IGR(200)
      COMMON /COMM1/ SL(10),C(200,10),NPER,JFLAG,SLACK(10)
      COMMON /COMM8/ B(10)
      COMMON /COMM9/ A(200,50),RHS(50),PER(50),EXT(50),JW(200),
     &NG,IPR(200)
      IF(IOFL.GT.1) GO TO 973
      OPEN(8,FORM="FORMATTED",ACCESS="SEQUENTIAL",MODE="OUT",
     &CARRIAGE=.TRUE.,FILE="FILEOUT")
973   WRITE(8,600)
      DO 398 K=1,NP
398   ACH(K)=0.0
      IF(IOFL.NE.1) GO TO 25
      WRITE(8,601) IOFL
      WRITE(8,602)
      WRITE(8,603)
      DO 400 I=1,JL
      II=JS(I)
      WRITE(8,604) JS(I),IGR(II),ILEV(II)
400   CONTINUE
      DO 418 J=1,NG
      EE=(-1.)*EXT(J)
      EJ=0.
      EI=AMAX1(EE,EJ)
      P2=PER(J)-EI
      L=IPR(J)
418   ACH(L)=ACH(L)+P2*JW(J)
      IF(LK.EQ.0) GO TO 777
      WRITE(8,605)
      WRITE(8,603)
      DO 401 I=1,LK
      II=CAND(I)
      WRITE(8,604) II,IGR(II),ILEV(II)
      DO 401 K=1,NP
      ACH(K)=ACH(K)+AH(II,K)
401   CONTINUE
      GO TO 777
25    IF(IOFL.NE.2) GO TO 50
      WRITE(8,601) IOFL
      WRITE(8,602)
      WRITE(8,603)
      DO 403 I=1,JL
      II=JS(I)
      WRITE(8,604) JS(I),IGR(II),ILEV(II)
403   CONTINUE
      DO 419 J=1,NG
      EE=(-1.)*EXT(J)
      EJ=0.
      EI=AMAX1(EE,EJ)
      P2=PER(J)-EI
      L=IPR(J)
419   ACH(L)=ACH(L)+P2*JW(J)
      GO TO 777
50    DO 405 K=1,NP
      IF(PROF(K).GT.0.) GO TO 794
405   CONTINUE
      GO TO 795
794   CONTINUE
      WRITE(8,601) IOFL
      WRITE(8,602)
      WRITE(8,603)
      DO 406 I=1,JL
      II=JS(I)
      IF(JS(I).EQ.0) GO TO 406
      WRITE(8,604) JS(I),IGR(II),ILEV(II)
      DO 407 K=1,NP
      ACH(K)=ACH(K)+AH(II,K)
407   CONTINUE
406   CONTINUE
777   WRITE(8,606)
      DO 402 K=1,NP
402   WRITE(8,607) K,ACH(K)
795   CONTINUE
      IF(IOFL.NE.3) GO TO 909
      DO 411 L=1,NPER
      CC=0.0
      DO 417 M=1,JL
      I=JS(M)
      CC=CC+C(I,L)
417   CONTINUE
      SLACK(L)=B(L)-CC
      WRITE(8,609) L,SLACK(L)
411   CONTINUE
909   CONTINUE
      RETURN
600   FORMAT('1',35X,'MULTIPLE OBJECTIVE RESOURCE ALLOCATION
     & SOLUTION SYSTEM OUTPUT')
601   FORMAT('1',55X,'OUTPUT FROM PHASE ',I1)
602   FORMAT('0',45X,'THE FOLLOWING VARIABLES ARE IN THE SOLUTION')
603   FORMAT('0',17X,'VARIABLE INDEX',17X,'ALTERNATIVE GROUP NUMBER',17X,
     &'FUNDING ALTERNATIVE NUMBER')
604   FORMAT(' ',23X,I3,30X,I3,38X,I2)
605   FORMAT('0',59X,'FITBACK VARIABLES')
606   FORMAT('0',15X,'ACHIEVEMENT VECTOR')
607   FORMAT(' ',20X,'PRIORITY 'I1,5X,F10.4)
609   FORMAT('0',40X,'SLACK REMAINING IN PERIOD ',I2,':',5X,F10.6)
      END
```

B. RAM/VM

```
C*********************** PROGRAM RAM/VM ******************************
C
C ****SUBROUTINE FUNCTIONS
C
C        1. FEAS--CHECKS AN EXCHANGE OPERATION FOR BUDGET FEASIBILITY.
C
C        2. COMPARE--CHECKS AN OPERATION FOR PROFITABILITY.
C
C        3. RERANK--REORDERS THE PROJECTS IN THE SETS JS AND NS AFTER
C           EACH EXCHANGE. (JS IS RANKED IN ASCENDING ORDER OF PROFIT,
C           NS IN DESCENDING ORDER)
C
C        4. FITBACK--USES UP BUDGET SLACK BY ADDING ANY FEASIBLE PROJECTS
C           (REGARDLESS OF RELATIVE PROFIT) TO JS.
C
C*******************************************************************
C
C ****INPUT VARIABLES
C
C        1. NVAR--THE NUMBER OF 0-1 VARIABLES.
C        2. NG--THE NUMBER OF OBJECTIVES.
C        3. NP--THE NUMBER OF PRIORITY LEVELS.
C        4. NPER--THE NUMBER OF BUDGET PERIODS.
C        5. NGR--THE NUMBER OF MUTUALLY EXCLUSIVE GROUPS (ALTERNATIVE GROUPS).
C        6. W(J)--THE WEIGHTING FACTOR ASSOCIATED WITH OBJECTIVE J.
C        7. PR(J)--THE PRIORITY LEVEL WHICH INCLUDES OBJECTIVE J.
C        8. IGR(I)--THE ALTERNATIVE GROUP WHICH CONTAINS VARIABLE I.
C        9. A(I)--THE CONTRIBUTION OF VARIABLE I TO THE ACHIEVEMENT OF
C           AN OBJECTIVE.
C       10. C(I,L)--THE COST OF VARIABLE I IN BUDGET PERIOD L.
C       11. B(L)--THE BUDGET UPPER BOUND FOR PERIOD L.
C       12. ILEV(I)--THE FUNDING LEVEL WITHIN IGR(I) REPRESENTED BY VARIABLE I.
C       13. IGS--IF SET TO 1, INDICATES THE PRESENCE OF ALTERNATIVE GROUPS
C            WHICH MUST BE FUNDED AT SOME NON-ZERO LEVEL.
C
C***********************************************************************
C
C****INTERNAL VARIABLES
C
C        1. R(I)--THE PROPORTION OF AVAILABLE FUNDS REQUIRED BY
C           VARIABLE I,AVERAGED OVER ALL BUDGET PERIODS.
C        2. AH(I,K)--FOR EACH VARIABLE I,THIS IS AN NP-DIMENSIONAL VECTOR,
C           WHOSE KTH COMPONENT IS A WEIGHTED SUM OF CONTRIBUTIONS OF VARIABLE I
C           TO THE OBJECTIVES AT PRIORITY LEVEL K.
C        3. INIF--A FLAG WHICH COMMUNICATES THE CURRENT STAGE TO THE
C           SUBROUTINE 'COMPARE'.
C        4. INC(I)--HAS A VALUE OF 1 IF VARIABLE I IS IN THE SOLUTION;
C           0 OTHERWISE.
C        5. IGR(M)--HAS A VALUE OF 1 IF THE GROUP M IS REPRESENTED IN
C           THE SOLUTION;0 OTHERWISE
C        6. JS(M)--TAKES ON THE INDEX NUMBER OF THE MTH VARIABLE
C           IN THE SOLUTION
C        7. JL--THE NUMBER OF VARIABLES CURRENTLY IN THE SOLUTION SET;JS(M).
C        8. NS(M)--TAKES ON THE INDEX NUMBER OF THE MTH VARIABLE NOT IN THE
C           SOLUTION.
C        9. LL--THE NUMBER OF VARIABLES CURRENTLY OUT OF THE SOLUTION.
C       10. SLACK(L)--A VECTOR WHICH INDICATES UNSPENT FUNDS IN
C           EACH BUDGET PERIOD FOR THE CURRENT SOLUTION.
C       11. NUMF--IS A FLAG WHICH INDICATES THE TYPE OF EXCHANGE
C           TO PERFORM.
```

```
C       12. PROF(K)--IS A NP-DIMENSIONAL VECTOR WHOSE KTH COMPONENT
C           IS THE PROFIT GAINED IN THE KTH PRIORITY LEVEL SO FAR IN
C           THE CURRENT EXCHANGE CYCLE.
C       13. PROFN(K)--IS A NP-DIMENSIONAL VECTOR WHOSE KTH COMPONENT IS THE
C           PROFIT IN THE KTH PRIORITY LEVEL FOR THE EXCHANGE
C           UNDER CONSIDERATION.
C       14. IFLAG--TAKES ON A VALUE OF 1 IF AN EXCHANGE IS PROFITABLE;
C           0 OTHERWISE.
C       15. JFLAG--TAKES ON A VALUE OF 1 IF AN EXCHANGE IS FEASIBLE; 0 OTHERWISE
C       16. CAND(I)--TAKES ON THE INDEX NUMBER OF THE ITH VARIABLE
C           IN THE FITBACK SET.
C
C*****************************************************************************
C
C****CURRENT DIMENSION STATUS
C
C           1. NUMBER OF GOALS: 200
C
C           2. NUMBER OF PRIORITY LEVELS: 3
C
C           3. NUMBER OF BUDGET PERIODS : 10
C
C           4. NUMBER OF 0-1 VARIABLES(ALTERNATIVES): 475
C
C           5. NUMBER OF ALTERNATIVE GROUPS: 150
C
C   ***************************************************************************
C
C              RAM/VM WAS DEVELOPED AND WRITTEN AT
C
C                    ANALYTIC SERVICES, INC
C                     (TACTICAL DIVISION)
C                     400 ARMY-NAVY DRIVE
C                   ARLINGTON, VIRGINIA  22202
C
C                       (703) 979-0700
C                        AV   225-5640
C
C   ***************************************************************************
      DIMENSION A(475),R(475)
      INTEGER OV,HV,OV1,OV2
      INTEGER PR(475),W(475)
      COMMON /COMM1/ SL(10),C(475,10),NPER,JFLAG,SLACK(10)
      COMMON /COMM2/ JS(150),JL
      COMMON /COMM3/ AH(475,3),PROF(3),IFLAG,NP,PROFN(3)
      COMMON /COMM4/ IGN(150),INC(475),CAND(150),NGR,NVAR
     &,LK,ILEV(200),ACH(3)
      COMMON /COMM5/ NUMF,OV,JV,KV,LV,MV,HV,IOUT1,IOUT2
     & ,IOUT3,IN1,IN2,IN3,NL,NM,NH,II,JJ,KK
     &,IA,IB,IC,IAA,IBB,ICC
      COMMON /COMM6/ NS(475),LL,IGR(200)
      COMMON /COMM7/ INIF
      COMMON /COMM8/ B(10)
      COMMON /COMM9/ MMLFL
      READ(10,501) NVAR,NG,NP,NPER,NGR,IGS
      READ(10,502) (W(J),PR(J),J =1,NG)
      DO 300 L=1,NPER
 300  READ(10,503) (C(I,L),I=1,NVAR)
      READ(10,502) (IGR(I),I=1,NVAR)
      READ(10,502) (ILEV(I),I = 1,NVAR)
      READ(10,503) (B(L),L=1,NPER)
 501  FORMAT(16I5)
 502  FORMAT(16I5)
 503  FORMAT(10F8.0)
 996  FORMAT(' ',25I5)
C
C**SECTION1.--ESTABLISH A GOOD INITIAL SOLUTION-CHOOSE MOST COST EFFECTIVE
C                SET OF OPTIONS UNTIL FUNDS ARE EXPENDED.
C
C**FIRST FIND THE AVERAGE PROPORTION OF AVAILABLE FUNDS USED BY EACH VARIABLE I
C
C   1. CALCULATE R(I)
C
      DO 1  I = 1,NVAR
      R(I)=0.0
      DO 2  L = 1,NPER
      R(I)=R(I)+(C(I,L)/B(L))
 2    CONTINUE
      R(I) = R(I)/NPER
 1    CONTINUE
C
C**COMPUTE (BENEFIT(S)/AVERAGE COST) VECTOR FOR ALL OPTIONS.
C
C  2. CALCULATE AH(I,K)
      DO 4 K=1,NP
      DO 4   I = 1,NVAR
      AH(I,K) = 0.0
 4    CONTINUE
      DO 3  J = 1,NG
      READ (10,503) (A(I),I = 1,NVAR)
      K=PR(J)
      DO 5  I = 1,NVAR
      IF(IGS.EQ.1.AND.K.EQ.1) GO TO 560
      GO TO 561
 560  AH(I,K)=A(I)
      GO TO 5
 561  CONTINUE
      AA = (A(I)*W(J))/R(I)
      AH(I,K) = AH(I,K) + AA
 5    CONTINUE
 3    CONTINUE
C
C**DETERMINE INITIAL SOLUTION.
C
C**ADD A DUMMY VARIABLE TO INSURE THAT 1:1 EXCHANGES WILL BE MADE
C  WITH LEAST BENEFICIAL OPTION.
C
      ISIZ=1
      IF(ISIZ.NE.0) GO TO 907
      NVAR=NVAR+1
      NGR=NGR+1
      DO 905 K=1,NP
      AH(NVAR,K)=0.
 905  CONTINUE
      IGR(NVAR)=NGR
      ILEV(NVAR)=1
      DO 906 L=1,NPER
      C(NVAR,L) = 1000.
 906  CONTINUE
 907  CONTINUE
```

```
C
C**NOW CHOOSE THE OPTION WITH THE LEXICOGRAPHIC MAXIMUM (BENEFIT/COST)
C  VECTOR FOR EACH SUBSET OF MUTUALLY EXCLUSIVE OPTIONS.
C
      INIF = 1
      LL = 0
      DO 6  M = 1,NGR
      MAX=0
      DO 7  I=1,NVAR
      IF(IGR(I) .NE. M) GO TO 7
      INC(I)=0
      IV=I
      IF(MAX.EQ.0) GO TO 101
      CALL COMPARE (IV,OV,INIF)
      IF(INC(I).EQ.0) GO TO 102
      LL=LL+1
      NS(LL)=OV
      OV = I
      GO TO 7
 102  LL = LL + 1
      NS(LL) = I
      GO TO 7
 101  OV=I
      INC(I)=1
      MAX=1
 7    CONTINUE
      JS(M)=OV
 6    CONTINUE
      INIF=0
C
C**NOW RANK THE SET CONTAINING THE MAXIMUM (BENEFIT/COST) VECTOR
C  OPTIONS FOR EACH ALTERNATIVE GROUP IN DESCENDING ORDER OF (BENEFIT/COST)
C  VECTOR MAGNITUDE.
C
      DO 8  M=2,NGR
      N=M
 103  MO=JS(N-1)
      MN=JS(N)
      CALL COMPARE(MN,MO,INIF)
      IF(IFLAG .EQ. 0) GO TO 8
      INTER = JS(N-1)
      JS(N-1)=JS(N)
      JS(N)=INTER
      N=N-1
      IF(N-1) 8,8,103
 8    CONTINUE
      JL=NGR
C
C**CHECK TO SEE IF THE INITIAL SOLUTION CONFORMS TO ALL FUNDING CONSTRAINTS
C  IF NOT, REMOVE ENTRY WITH SMALLEST BENEFIT VALUE AND PLACE IN SET OF
C  NON-FUNDED OPTIONS--REPEAT UNTIL BUDGET FEASIBILITY IS REACHED.
C
C***CHECK (JS) FOR FEASIBILITY AND DELETE BOTTOM ENTRY IF NOT FEASIBLE
C
 104  DO 9 L = 1,NPER
      CC=0.0
      DO 10 M = 1,JL
      I=JS(M)
      CC=CC+C(I,L)
      IF(CC.GT.B(L)) GO TO 105
 10    CONTINUE
       SLACK(L)=B(L)-CC
 9     CONTINUE
       GO TO 106
 105   II = JS(JL)
       INC(II)=0
       JL=JL-1
       LL=LL+1
       NS(LL)=II
       GO TO 104
C
C**EXCEPT FOR A PRIORITY ONE GOAL INDICATING THOSE SUBSETS WHICH MUST
C  RECEIVE NON-ZERO FUNDING (IF PRESENT), RERANK OPTIONS IN BOTH SETS
C  (IN THE SOLUTION AND OUT OF THE SOLUTION) IN RESPECTIVELY, ASCENDING
C  AND DESCENDING ORDER OF BENEFIT VALUE.
C
 106   DO 11 K = 1,NP
       DO 11 I = 1,NVAR
       IF(IGS.EQ.1.AND.K.EQ.1) GO TO 11
       AH(I,K)=AH(I,K)*R(I)
 11    CONTINUE
       CALL RERANK
 500   CONTINUE
       KKFL=0
C
C**NOW USING ANY SLACK FUNDS CHECK THOSE SUBSETS OF MUTUALLY EXCLUSIVE
C  OPTIONS WHICH ARE NOT IN THE INITIAL SOLUTION TO SEE IF _A_N_Y MEMBER
C  CAN BE FIT BACK INTO THE INITIAL SOLUTION.  THESE ARE IDENTIFIED IN
C  THE OUTPUT AS "FITBACK VARIABLES" IF PRESENT.
C
       CALL FITBACK(KKFL)
C
C**NOW OUTPUT INITIAL SOLUTION.
C
       IOFL=1
       CALL OUTPUT(IOFL)
C
C*******SECTION 2    FIRST EXCHANGE*********
C
C**IN THIS SECTION 2:1 AND 1:1 EXCHANGES ARE MADE BETWEEN THE JS (FUNDED) SET
C  AND THE NS (NOT FUNDED) SET.
C
C**NOW TEST IF THE LAST 2 MEMBERS OF THE RANKED NS LIST BELONG TO THE SAME
C  MUTUALLY EXCLUSIVE GROUP.  IF SO--SET MMFL TO THE INDEX NO. OF THAT GROUP
C
       MMLFL = 0
       JE=NS(LL)
       LM1=LL-1
       JF=NS(LM1)
       IF(IGR(JE).EQ.IGR(JF)) MMLFL = IGR(JE)
       IIJ=0
 116   CONTINUE
       IIJ=IIJ+1
C
C**ZERO OUT EXCHANGE PROFITABILITY INDICATOR.
C
       DO 12 K = 1,NP
       PROF(K)=0.0
 12    CONTINUE
C
```

```
C**NOW SET UP 2:1 EXCHANGE
C
      NV2=NVAR+2
      INIF=3
      DO 13 NL=1,LL
      IDUM=0
      NH=NL+1
      JJFL=1
C
C**IF LL LIST IS EXHAUSTED--EXIT EXCHANGE CYCLE.
C
 109  IF(NH.GT.LL) GO TO 13
      HV = NS(NH)
      LV=NS(NL)
C
C**TWO ENTERING ALTERNATIVES MAY NOT BELONG TO THE SAME
C  MUTUALLY EXCLUSIVE GROUP.
C
      IF(IGR(HV) .EQ. IGR(LV)) GO TO 107
      GO TO 108
 107  NH=NH+1
      GO TO 109
 108  CONTINUE
C
C**DETERMINE COMBINED CONTRIBUTION TO ALL GOALS J, OF
C  TWO "ENTERING" ALTERNATIVES.
C
      DO 14 J=1,NP
      AH(NV2,J)=AH(HV,J)+AH(LV,J)
 14   CONTINUE
C
C**DETERMINE COMBINED COST IN ALL BUDGET PERIODS L, OF
C  TWO "ENTERING" ALTERNATIVES.
C
      DO 15 L=1,NPER
      C(NV2,L)=C(HV,L)+C(LV,L)
 15   CONTINUE
C
C**NOW LOOK FOR 2:1 EXCHANGES WITH MEMBERS OF JS SET.
C
      DO 16 II=1,JL
      OV=JS(II)
      IS=IGR(LV)
      IX=IGR(HV)
C
C**EXCHANGE MUST BE FEASIBLE WITH RESPECT TO MUTUALLY
C  EXCLUSIVE SUBSETS.
C
      IF(IGN(IS).NE.O.AND.IGR(LV).NE.IGR(OV)) GO TO 16
      IF(IGN(IX).NE.O.AND.IGR(HV).NE.IGR(OV)) GO TO 110
      NV=NV2
C
C**TEST EXCHANGE FOR PROFITABILITY (MUST BE MORE PROFITABLE
C  THAN ANY OTHER FEASIBLE EXCHANGE TRIED IN 2:1 CYCLE).
C
      CALL COMPARE(NV,OV,INIF)
      IF(IFLAG.EQ.O) GO TO 111
C
C**TEST EXCHANGE FOR BUDGET FEASIBILITY.
C
```

```
      CALL FEAS(OV,NV)
      IF(JFLAG.EQ.0) GO TO 110
      NUMF =2
      GO TO 112
C
C**IF ANY 2:1 EXCHANGE WITH HIGHER RANKING OF 2 "ENTERING"
C  VARIABLES HAS BEEN UNPROFITABLE, OR IF ALL 1:1 EXCHANGES
C  WITH HIGHER RANKING OF 2 "ENTERING" VARIABLES HAVE BEEN
C  UNPROFITABLE, SKIP 1:1 EXCHANGE.
C
 110  IF(JJFL.GT.1) GO TO 16
      IF(IDUM.EQ.LV) GO TO 16
      NV=LV
C
C**ASSUME 1:1 EXCHANGE FEASIBILITY WITH RESPECT TO MUTUALLY EXCLUSIVE
C GROUPS.
C
      IF(NV.EQ.LNM.AND.OV.EQ.LNN) GO TO 16
C
C**CHECK 1:1 EXCHANGE FOR PROFITABILITY AND BUDGET FEASIBILITY.
C
      CALL COMPARE(NV,OV,INIF)
      LNM=NV
      LNN=OV
      IF(IFLAG.EQ.0) GO TO 16
      CALL FEAS(OV,NV)
      IF(JFLAG .EQ. 0) GO TO 16
      NUMF=1
      GO TO 112
C
C**AFTER NON-PROFITABLE 2:1 EXCHANGE HAS BEEN TRIED--IF INDEX OF HIGH
C  RANKING "ENTERING" VARIABLE IS 1, OR IF HIGH RANKING "ENTERING"
C  VARIABLE'S MUTUALLY EXCLUSIVE GROUP IS REPRESENTED IN THE FUNDED
C  SET, INCREMENT INDEX OF HIGH RANKING "ENTERING" VARIABLE AND
C  CONTINUE.  IF NOT, INCREMENT INDEX OF LOWER RANKING "ENTERING"
C  VARIABLE AND CONTINUE
C
 111  IF(I1).EQ.1) GO TO 13
      IF(IGN(IS).EQ.1) GO TO 13
      NH=NH+1
      JJFL=2
      GO TO 109
C
C**IF A PROFITABLE AND FEASIBLE EXCHANGE WAS FOUND, CHANGE CRITERION
C  FOR PROFITABILITY, AND SET UP PROVISIONAL EXCHANGE VARIABLES.
C
 112  DO 17 K = 1,NP
      PROF(K)=PROFN(K)
 17   CONTINUE
      IA = I1
      IOUT=OV
      IN1 = LV
      IAA = NL
      IF(NUMF.NE.2) GO TO 113
      IN2 = HV
      IBB = NH
      GO TO 111
 113  IN2 = 0
 16   CONTINUE
      IDUM=LV
      GO TO 107
 13   CONTINUE
      DO 18 K=1,NP
C
C**IF NO PROFITABLE AND FEASIBLE EXCHANGE WAS FOUND--FITBACK USING
C  REMAINING SLACK AND OUTPUT SECOND STAGE SOLUTION.
C
      IF (PROF(K).GT.0.0) GO TO 114
 18   CONTINUE
      KKFL = 1
      CALL FITBACK(KKFL)
      IOFL=2
      CALL OUTPUT(IOFL)
      IF(ISIZ.EQ.1) GO TO 810
C
C**MAKE MOST PROFITABLE FEASIBLE EXCHANGE FOUND AND BEGIN CYCLE AGAIN.
C
 114  INC(IOUT)=0
      INC(IN1)=1
      INTER3=JS(IA)
      JS(IA)=NS(IAA)
      N=JS(IA)
      IJ=IGR(N)
      NS(IAA)=INTER3
      IP=IGR(INTER3)
      IGN(IP)=0
      IGN(IJ)=1
      IF(IN2.EQ.0) GO TO 115
      LLL=0
      LL=LL-1
      DO 396 I=1,LL
      J=I+1
      IF(NS(I).EQ.IN2) LLL=1
      IF(LLL.EQ.1) NS(I)=NS(J)
 396  CONTINUE
      INC(IN2)=1
      JL=JL+1
      JS(JL)=IN2
      LX=IGR(IN2)
      IGN(LX)=1
 115  CALL RERANK
      DO 28 L=1,NPER
      CC=0.0
      DO 29 M=1,JL
      I=JS(M)
      CC=CC+C(I,L)
 29   CONTINUE
      SLACK(L)=B(L)-CC
 28   CONTINUE
      GO TO 116
 810  IOFL=3
      CALL OUTPUT(IOFL)
      CLOSE(8)
      CLOSE(10)
      STOP
      END
C
C*****SUBROUTINES*****************************
C
      SUBROUTINE FEAS(LOV,NV)
```

```
C
C**CHECKS FOR BUDGET FEASIBILITY.
C
      COMMON /COMM1/ SL(10),C(475,10),NPER,JFLAG,SLACK(10)
      COMMON /COMM7/ INIF
      DO 1 L=1,NPER
      SL(L) = SLACK(L)
      SL(L)=SL(L)+C(LOV,L)-C(NV,L)
      SLF=SL(L)+.0005
      IF(SLF) 10,1,1
 1    CONTINUE
      JFLAG=1
      GO TO 11
 10   JFLAG=0
 11   RETURN
      END
C
C********************************************
C
      SUBROUTINE RERANK
C
C**REORDERS PROJECTS IN JS AND NS AFTER EACH EXCHANGE.
C
      COMMON /COMM2/ JS(150),JL
      COMMON /COMM7/ INIF
      COMMON /COMM3/ AH(475,3),PROF(3),IFLAG,NP,PROFN(3)
      COMMON /COMM6/ NS(475),LL,IGR(200)
      INIF = 0
      IF(JL.LT.2) GO TO 71
C***********************(JS)**************
      DO 70 M=2,JL
      N=M
 700  MO=JS(N-1)
      MN=JS(N)
      CALL COMPARE(MN,MO,INIF)
      IF(IFLAG.EQ.1) GO TO 70
      INTER=JS(N-1)
      JS(N-1)=JS(N)
      JS(N)=INTER
      N=N-1
      IF(N-1) 70,70,700
 70   CONTINUE
 71   CONTINUE
C***********************(NS)**************
      DO 57 M = 2,LL
      N=M
 604  MO=NS(N-1)
      MN=NS(N)
      CALL COMPARE(MN,MO,INIF)
      IF(IFLAG.EQ.0) GO TO 57
      INTER=NS(N-1)
      NS(N-1) = NS(N)
      NS(N) = INTER
      N=N-1
      IF(N-1) 57,57,604
 57   CONTINUE
      RETURN
      END
C
C********************************************
```

```
C
      SUBROUTINE FITBACK (KKFL)
C
C**USES UP BUDGET SLACK TO FIT BACK PROJECTS.
C
      COMMON /COMM1/ SL(10),C(475,10),NPER,JFLAG,SLACK(10)
      COMMON /COMM2/ JS(150),JL
      COMMON /COMM3/ AH(475,3),PROF(3),IFLAG,NP,PROFN(3)
      COMMON /COMM4/ IGN(150),INC(475),CAND(150),NGR,NVAR
     &,LK,ILEV(200),ACH(3)
      COMMON /COMM6/ NS(475),LL,IGR(200)
      COMMON /COMM7/ INIF
      DO 15 L=1,NPER
      SL(L)=SLACK(L)
 15   CONTINUE
      DO 11 M=1,NGR
      IGN(M)=0
 11   CONTINUE
      DO 10 I=1,JL
      N=JS(I)
      M=IGR(N)
      IGN(M)=1
 10   CONTINUE
      LK=0
      DO 12 I=1,LL
      II=NS(I)
      MP=0
      DO 19 K=1,NP
      MP=MP+AH(II,K)
 19   CONTINUE
      IF(MP.EQ.0) GO TO 12
      M=IGR(II)
      IF(IGN(M).NE.0) GO TO 12
      DO 13 L=1,NPER
      IF(C(II,L).GT.SL(L)) GO TO 12
 13   CONTINUE
      LK=LK+1
      CAND(LK)=II
      DO 14 L=1,NPER
      SL(L)=SL(L)-C(II,L)
 14   CONTINUE
 12   CONTINUE
      IF(KKFL .EQ. 0 .OR. LK .EQ. 0) GO TO 17
      DO 18 I = 1,LK
      JJ = CAND(I)
      INC(JJ) =1
      JL = JL + 1
      LL = LL-1
      JS(JL) = JJ
      JK = IGR(JJ)
      IGN(JK) =1
 18   CONTINUE
 17   CONTINUE
      RETURN
      END
C
C**********************************************
C
      SUBROUTINE COMPARE(IN,IO,INIF)
C
```

```
C**CHECKS FOR PROFITABILITY.
C
      COMMON /COMM3/ AH(475,3),PROF(3),IFLAG,NP,PROFN(3)
      COMMON /COMM4/ IGN(150),INC(475),CAND(150),NGR,NVAR
     &,LK,ILEV(200),ACH(3)
      COMMON /COMM2/ JS(150),JL
      IF(INIF .NE. 1) GO TO 11
      DO 1 K = 1,NP
      IF(AH(IN,K) - AH(IO,K)) 2,1,3
 1    CONTINUE
 2    GO TO 10
 3    INC(IN) = 1
      INC(IO) = 0
      GO TO 10
 11   IF(INIF .NE. 0) GO TO 12
      DO 4 K = 1,NP
      IF(AH(IN,K) - AH(IO,K)) 5,4,6
 4    CONTINUE
 5    IFLAG = 0
      GO TO 10
 6    IFLAG = 1
      GO TO 10
 12   IF(INIF .NE. 3) GO TO 10
      DO 7 K = 1,NP
      PROFN(K) = AH(IN,K)-AH(IO,K)
 7    CONTINUE
      DO 8 K = 1,NP
      IF(PROFN(K)-PROF(K)) 18,8,20
 8    CONTINUE
 18   IFLAG = 0
      GO TO 10
 20   IFLAG = 1
 10   RETURN
      END
C
C**************************************************
C
      SUBROUTINE OUTPUT(IOFL)
C
C**WRITES OUT THE SOLUTION.
C
      COMMON /COMM4/ IGN(150),INC(475),CAND(150),NGR,NVAR
     &,LK,ILEV(200),ACH(3)
      COMMON /COMM2/ JS(150),JL
      COMMON /COMM3/ AH(475,3),PROF(3),IFLAG,NP,PROFN(3)
      COMMON /COMM6/ NS(475),LL,IGR(200)
      COMMON /COMM1/ SL(10),C(475,10),NPER,JFLAG,SLACK(10)
      COMMON /COMM8/ B(10)
      COMMON /COMM9/ MMLFL
      IF(IOFL.GT.1) GO TO 973
      OPEN(8,FORM="FORMATTED",ACCESS="SEQUENTIAL",MODE="OUT",
     &CARRIAGE=.TRUE.,FILE="FILEOUT")
 973  WRITE(8,600)
      DO 398 K=1,NP
 398  ACH(K)=0.0
      IF(IOFL.NE 1) GO TO 25
      WRITE(8,601) IOFL
      WRITE(8,602)
      WRITE(8,603)
      DO 400 I=1,JL
      II=JS(I)
      WRITE(8,604) JS(I),IGR(II),ILEV(II)
      DO 400 K=1,NP
      ACH(K)=ACH(K)+AH(II,K)
400   CONTINUE
      IF(LK.EQ.0) GO TO 777
      WRITE(8,605)
      WRITE(8,603)
      DO 401 I=1,LK
      II=CAND(I)
      WRITE(8,604) II,IGR(II),ILEV(II)
      DO 401 K=1,NP
      ACH(K)=ACH(K)+AH(II,K)
401   CONTINUE
      GO TO 777
25    IF(IOFL.NE.2) GO TO 50
      WRITE(8,601) IOFL
      WRITE(8,602)
      WRITE(8,603)
      DO 403 I=1,JL
      II=JS(I)
      WRITE(8,604) JS(I),IGR(II),ILEV(II)
      DO 403 K=1,NP
      ACH(K)=ACH(K)+AH(II,K)
403   CONTINUE
      GO TO 777
50    DO 405 K=1,NP
      IF(PROF(K).GT.0.) GO TO 794
      GO TO 795
794   CONTINUE
405   CONTINUE
      WRITE(8,601) IOFL
      WRITE(8,602)
      WRITE(8,603)
      DO 406 I=1,JL
      II=JS(I)
      IF(JS(I).EQ.0) GO TO 406
      WRITE(8,604) JS(I),IGR(II),ILEV(II)
      DO 407 K=1,NP
      ACH(K)=ACH(K)+AH(II,K)
407   CONTINUE
406   CONTINUE
777   WRITE(8,606)
      DO 402 K=1,NP
402   WRITE(8,607) K,ACH(K)
795   CONTINUE
      IF(IOFL.NE.3) GO TO 909
      DO 411 L=1,NPER
      CC=0.0
      DO 417 M=1,JL
      I=JS(M)
      CC=CC+C(I,L)
417   CONTINUE
      SLACK(L)=B(L)-CC
      WRITE(8,609) L,SLACK(L)
411   CONTINUE
909   CONTINUE
      RETURN
600   FORMAT('1',35X,'MULTIPLE OBJECTIVE RESOURCE ALLOCATION
     & SOLUTION SYSTEM OUTPUT')
```

```
601  FORMAT('1',55X,'OUTPUT FROM PHASE ',I1)
602  FORMAT('0',45X,'THE FOLLOWING VARIABLES ARE IN THE SOLUTION')
603  FORMAT('0',17X,'VARIABLE INDEX',16X,'ALTERNATIVE GROUP NUMBER',16X,
    &'FUNDING ALTERNATIVE NUMBER')
604  FORMAT(' ',23X,I3,30X,I3,38X,I2)
605  FORMAT('0',59X,'FITBACK VARIABLES')
606  FORMAT('0',15X,'ACHIEVEMENT VECTOR')
607  FORMAT(' ',20X,'PRIORITY 'I1,5X,F20.4)
609  FORMAT('0',40X,'SLACK REMAINING IN PERIOD ',I2,':',5X,F20.6)
     END
```

RESOURCE ALLOCATION METHODOLOGY FOR AIR FORCE R&D PLANNING

Volume 3: Guide to the Interactive *RAM* Program

CONTENTS
Volume 3

I. INTRODUCTION

This volume is one of four that document ANSER's development of R&D resource allocation methodology (RAM) for the Director of Program Integration, AF/RDX. Volume 1 provides an overview of the work and its applications. Volume 2 describes the RAM technique and how to use the general-purpose computer programs that incorporate it. Volume 3 describes how to use the interactive computer program developed for use of the RAM within AF/RDX, and Volume 4 describes the way in which we tested its computational performance. Each volume emphasizes some particular aspect of our research and can be read independently of the others.

This volume describes the computer software we wrote to enable AF/RDX to use the RAM algorithm with its in-house data base of Air Force R&D programs. It is designed for that computer system, specific hardware, and data base; consequently, the software is not readily transferrable to other systems. Nevertheless, other readers may also find the software design potentially useful for decision making in resource allocation.

Our goal in producing the software was to demonstrate the relative ease with which it could be used to obtain an investment strategy. The user of the software needs only a basic knowledge of computer programming. By merely specifying on the computer terminal the available resources as a function of time, the user receives from RAM the list of programs providing the greatest benefit. (This presumes, of course, that the program alternatives have been evaluated in advance with respect to the chosen objectives.) To determine the impact of funding cuts or enhancements, the user simply has to type in some data. One of the available outputs is a graphic display of the benefits and costs for the various investment strategies generated. This display enables the user to see whether the strategies are at "efficient" points on the cost/benefit curve.

In Chapter II, we describe the interactive portion of the software. Using the software does not require an understanding of RAM, but if interested, the reader may refer to Volume 2 for an explanation. The user may desire to make other parts of the data base interactive or to produce graphic displays other than those we have designed. Chapter III, which serves as a programmer's guide for those modifications, requires detailed knowledge of computer programming. Chapter IV contains the appropriate computer codes.

At the time this work was completed, the anticipated "benefit" data were not available in the official data base. Consequently, to demonstrate the software, we substituted a simple procedure that generates random benefit data. If appropriate data became available, this procedure could be easily changed using the information in Chapters III and IV.

II. USER'S GUIDES

A. User's Guide to the Program NOW

1. Background

NOW* is a computer program written in the PL/1 programming language;† it is on file at the Air Force Data Services Center's Multics computer system. NOW is the master program driving a set of subroutines, the purpose of which is to determine and compare investment strategies for the selection of funded program element (PE) alternatives within a mission area.

The information necessary for the execution of the program is part of a data base managed by AF/RDX on the Multics. The user must obtain the required Multics clearances to access the data base.

The NOW user should be aware of the general nature of the R&D data base. The data are grouped into 10 mission areas, indicated by the number codes 000, 100, 200, 300, 400, 500, 600, 700, 800, and 900. An eleventh category, "off," is also listed. The "off" category identifies the official data base, which contains official data in the 10 mission areas. The official data base contrasts with a working-level data base that, at any given time, may be in a state of flux. The "off" data base contains all R&D program elements. Users of NOW will normally access the official R&D data base. A

*Names of computer terms in this report are in upper case except where they are to be literal input to the computer system. Such inputs, when described in the test, are given in quotes.

†With FORTRAN language subroutines.

second data base, the "sc" data base, also exists, but, at the time of this writing, is not in use.

2. Program-User Interaction

The following is a step-by-step procedure for use of the interactive portion of the program. (See Figure 1 for interactions in an actual run.)

You* are now assumed to be within the Multics facility and to have a copy of program NOW. You must now log on to the system at a SECRET level. If your name is Smith, the appropriate log-on format is as follows:

```
"l Smith -auth s"
```

Initiate execution of program NOW by entering the word "now" via the terminal keyboard. The system will respond by requesting your Multics personal ID. For Smith, the appropriate response is "Smith," not "Smith -auth s".

After entry of your personal ID, the program asks a series of questions intended to determine your inputs to a resource allocation algorithm. The program first asks whether you desire the rd or sc data base. Respond by entering "rd" or "sc". In response, the program lists the codes indicating the mission areas to which you have clearance to access. The mission areas corresponding to the number codes are as follows:

000	Programs Not Assigned to a Mission Area
100	Strategic Offense

* For simplicity, "you" is synonomous with user throughout the rest of this chapter.

FIGURE 1
PROGRAM-USER INTERACTION

```
now

Enter your Multics person_id Clary

What database? rd or sc? rd

You have access to the following  mission areas:
000
100
200
300
400
500
600
700
800
900
off

off data? yes

Which mission area? if none, type 'none'. 100

What appropriation? 3600

How many years of cost would you like to consider? 7

Enter year, budget 79 1000000

Enter year, budget 80 100000

Enter year, budget 81 100000

Enter year, budget 82 100000

Enter year, budget 83 100000

Enter year, budget 84 100000

Enter year, budget 85 100000
```

200	Strategic Defense
300	Tactical Air Warfare
400	Space Launch and Orbital Support
500	Mobility
600	Reconnaissance
700	Command, Control and Communications
800	Technology Base
900	Defense-Wide Management and Support.

The program next asks whether you wish to use official data with the query

"off data?"

It also asks you to select a mission area. Respond with one of the mission area codes to which you have access. This selection determines which mission area the resource allocation routine will consider. If you wish to terminate the program, enter the word "none".

The R&D allocation process works with the following four appropriation categories and their respective number codes:

- o Aircraft Procurement, 3010
- o Missile Procurement, 3020
- o Other Procurement, 3080
- o RDT&E, 3600.

Each program element is stored in the data base with, among other things, its associated appropriation code (3010, 3020, 3080, or 3600). Thus, when you select an "appropriation," a subset of the program elements from the selected mission area is designated for input to the resource allocation program. You must respond to the query: "what appropriation?", with one of the four number codes. At the present time, data exist for the 3600 appropriation only.

You must now decide which budget years you want the resource allocation routine to consider and the budget levels for the selected years. For budget years, you may designate any set of years from 79, 80, 81, 82, 83, 84, and 85. The system will ask: "How many years of cost would you like to consider?" The system will then ask you to enter the budget year and budget level for the selected years. You should leave a space between the budget year and level. Note that the budget is entered in thousands of dollars; therefore, enter $100,000, for example, as "100".

You have now completed the necessary inputs for one execution of the resource allocation program. Table 1 gives a complete listing of your input alternatives.

TABLE 1
USER INPUT ALTERNATIVES

Data Base*	Mission Area*	Appropriation*	Budget Years†	Budget Levels
sc	000	3600	79	Expressed in thousands of dollars
rd	100	3080	80	
	200	3010	81	
	300	3020	82	
	400		83	
	500		84	
	600		85	
	700			
	800			
	900			

*Choose only one.
†Choose any or all.

At this point, before execution of the resource allocation algorithm, the system scans all PE alternatives. Those alternatives having zero funding levels for the specified budget years are not considered in the resource allocation algorithm. They are, however, printed out for your convenience.

After execution of the resource allocation algorithm, the system displays those program elements that the resource allocation algorithm selected for funding. It includes some self-explanatory, associated data base information. Costs of each PE alternative for each of the 7 budget years and benefits to the goals are also provided.

Note the instructions printed out just before the first page of classified output. After each page of classified output is printed, you must enter some character, for example, "a". Then, if you are using a CRT terminal and desire a permanent output record, you must make a copy on the Tektronix 4631 Photocopier. You must then reset the page and, finally, press the return key. The photocopying and page reset steps are, of course, omitted if you are at a hardcopy terminal. The instructions are printed out only once, but you must execute this sequence after each page of classified material is printed.

The next output is a summary of the costs and benefits of the set of program elements selected by the resource allocation subroutine. The total cost of the selected program elements summed over all cost years and the total benefits for each task* are printed out. An example of such output follows.

*"Benefits for each task" means the contributions toward achievement of the goals.

```
The total cost of the solution is     9700.00
      benefit to task  1 =    268.81
      benefit to task  2 =    230.89
      benefit to task  3 =    262.54
      benefit to task  4 =    181.86
Do you want the output from this strategy saved for graphs?  Please answer yes or no.
yes
do you want to generate more data before constructing graphs?
yes

Select one of the following:
1=different database
2=different mission area
3=different appropriation
4=different budgets for years currently considered
5=different years of cost data
0=stop)

What database? rd or sc? rd

You have access to the following  mission areas:
000
100
200
300
400
500
600
700
800
900
off
```

You must now decide whether you want the results of the resource allocation subroutine to be shown graphically. The system will ask, "Do you want the output from this strategy saved for graphs?" If the answer is "yes," you must then decide whether to generate additional results before constructing graphs or to proceed into the graphics segment of the program with the solutions you already have.

The system will ask, "Do you want to generate more data before constructing graphs?" If your answer is "yes," the system will display a set of alternatives for the next resource allocation. If your answer is "no," the system will take you through subroutine GRAPH before you can try any additional resource allocation strategies. (See Section II.B for subroutine GRAPH documentations.)

At some point, you must decide on your next resource allocation strategy. You are offered six alternatives: to use a different data base; to use a different mission area within your current data base; to use a different appropriation within your current data base and mission area; to use different budget levels; to use different budget years; or to exit the program. You need only enter the number designating the desired option.

If you do select the sixth alternative (exit the program), then further user/program interaction will consist of repetitions of previous interactions.

B. User's Guide to Subroutine GRAPH

This subroutine enables you to obtain graphics output that depicts the impact of alternative investment strategies on the costs and benefits pertaining to a single mission area.

We assume in this discussion that subroutine GRAPH is being called by the NOW program. Use of subroutine GRAPH under other circumstances is briefly discussed at the end of this section.

Within the context of the NOW driver program, subroutine GRAPH is automatically called after each execution of the resource allocation algorithm, RAM/VM*. Subroutine GRAPH presents you with a summary of the allocation results, such as the following:

*See Volume 2 for explanation of RAM/VM.

```
The total cost of the solution is 83.90
          benefit to task  1 = 110.00
          benefit to task  2 = 330.00
          benefit to task  3 =  30.58
          benefit to task  4 =   2219.00
          benefit to task  5 = 700.01
          benefit to task  6 =   2000.00
          benefit to task  7 =   4.60
          benefit to task  8 =   4.60
```

You are then asked if you wish to save the results to be plotted on graphs. (The information saved is, of course, considerably more extensive than what is shown.) If you do not want to save the results you are returned to the main program. If you do want to save the results, you are then asked if you want to generate more data before constructing graphs:

```
The total cost of the solution is 83.90
     contribution to benefit measure  1 =    110.00
     contribution to benefit measure  2 =    330.00
     contribution to benefit measure  3 =     30.58
     contribution to benefit measure  4 =   2219.00
     contribution to benefit measure  5 =    700.01
     contribution to benefit measure  6 =   2000.00
     contribution to benefit measure  7 =      4.60
     contribution to benefit measure  8 =      4.60
Do you want the output from this strategy saved for graphs?  Please answer yes or no.
yes
do you want to generate more data before constructing graphs?
yes
```

If you answer "yes," control is passed back to NOW for another resource allocation with a different investment strategy. If you answer "no," the subroutine enters the section where it constructs the graphs.

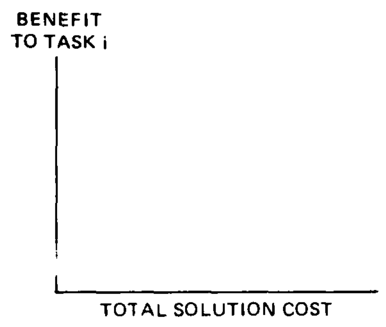
As described in Figure 2, four types of graphs are available through subroutine GRAPH. Note that Graph Type 4 can portray the results of only one resource allocation at a time. The graph will still represent only those results even if you have saved the results of applying resource allocation methodology to different mission areas or in some other manner have constructed noncomparable results in a series of resource allocations. The rest of the graphs compare the data resulting

FIGURE 2
SUMMARY OF SUBROUTINE GRAPH CAPABILITIES

Graph Type 1

Graph of benefit of solution to individual task versus total solution costs summed over all years.

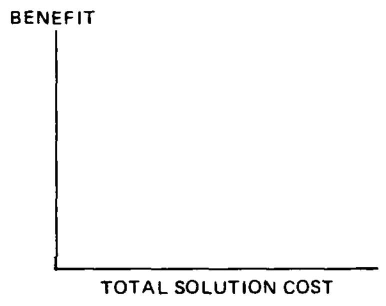
Each point plotted is total cost of solution due to strategy k versus benefit of that solution to task i, k = 1, 2, ..., j.



Graph Type 2

Total benefit versus total solution cost for each strategy.

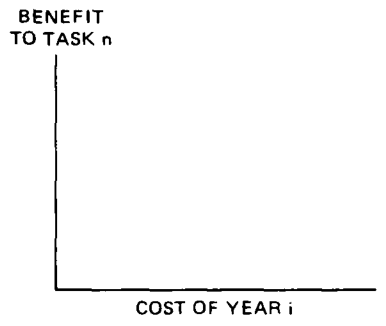
Each point plotted is the total cost of solution due to strategy k versus total benefit due to strategy k, k = 1, 2, ..., j.



Graph Type 3

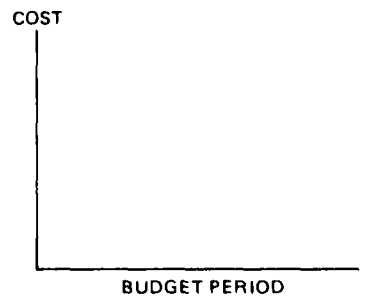
Individual task benefits versus cost of individual budget periods.

Each point plotted is the cost of year i versus the benefit to task n for the solution due to strategy k, k = 1, 2, ..., j.



Graph Type 4

Cost of a given strategy for each budget period.



from separate resource allocation runs. Clearly, you must make certain that inputs to such graphs are comparable.

Graph Type 1 depicts the total solution cost summed over all years versus the contribution of the solution to a specific task for each strategy. Thus, what is graphed is the set of points {(x(k),y(i,k), k=1,...,j}, where j is the total number of solutions saved for graphing; x(k) is the total cost of the solution corresponding to the kth strategy the user has saved; and y(i,k) is the benefit of the kth solution to the ith task. Each point on this graph represents a different resource allocation. You must input task index i. The computer/user interaction needed to generate this graph is:

```
g
Do you want any graphs of total solution costs summed over all years vs. contri
\cbution of solution to individue  benefit measures?
yes
Key in index for the benefit measure.
4
solution cost data            83.90         70.50        103.60
solution benefit data       2219.00       2035.00       1520.00
```

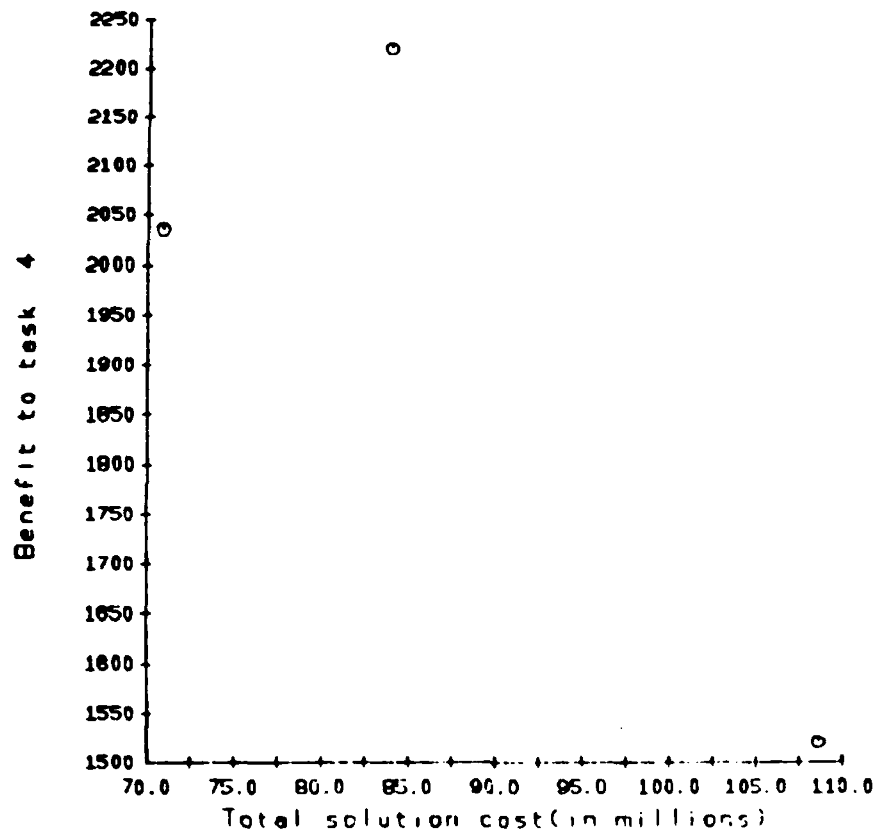
You can generate this graph for as many tasks as you desire. Figure 3 shows a sample of this graph type.

Graph Type 2 represents total solution benefits versus total solution costs for each strategy. Thus, the plot consists of the set of points {x(k), y(ngl,k), k=1,...,j}. The variables x(k) and j have the same definitions as those in Graph Type 1. The variable y(ngl,k) stores the sum of the benefits of the kth strategy, i.e.,

$$y(ngl,k) = \sum_{m=1}^{ng} y(m,k), \quad ng = \text{number of tasks.}$$

Once again, each point represents a different solution. The computer/user interaction to generate this graph is:

FIGURE 3
SAMPLE GRAPH TYPE 1



```
Do you want a graph of total benefit vs. total cost for each strategy?
yes
solution cost data            83.90   70.90      168.60
solution benefit data       5398.79 6066.10      1599.35
```

A sample of this graph type is given in Figure 4.

Graph Type 3 depicts the solution contribution to task n versus the cost of an individual budget period i for each strategy or solution. The plot consists of the set of points $\{(yr(i,k),\ y(n,k)),\ k=1,...,j\}$. The variables j and y(n,k) have the same definitions as those previously mentioned. The variable yr(i,k) is the total cost of the kth saved solution in the ith budget period. Again, each point represents a different strategy. The user/computer interaction necessary to generate this type of graph is:

```
Do you want any graphs of cost of individual budget periods vs. individual task benefits?
yes
Key in budget period desired.
3
Key in task benefit index.
5
To be plotted is a a graph of cost of year  3 vs. benefit to task  5 .
asolution cost data       36.10          33.50    32.70
solution benefit data       700.01    0.70         0.00
```

You can generate this graph type for as many budget periods and tasks as you desire. An example is given in Figure 5.

Graph Type 4 depicts the cost of a single strategy for each budget period. The results of each strategy are graphed separately to avoid the possibility of an inconsistent graph. The computer/user interaction necessary to generate this graph is:

```
Do you want graphs showing cost of strategy for each budget period
yes
```

The subroutine automatically generates this graph for each strategy as currently written. An example of this graph type is given in Figure 6.

FIGURE 4
SAMPLE GRAPH TYPE 2

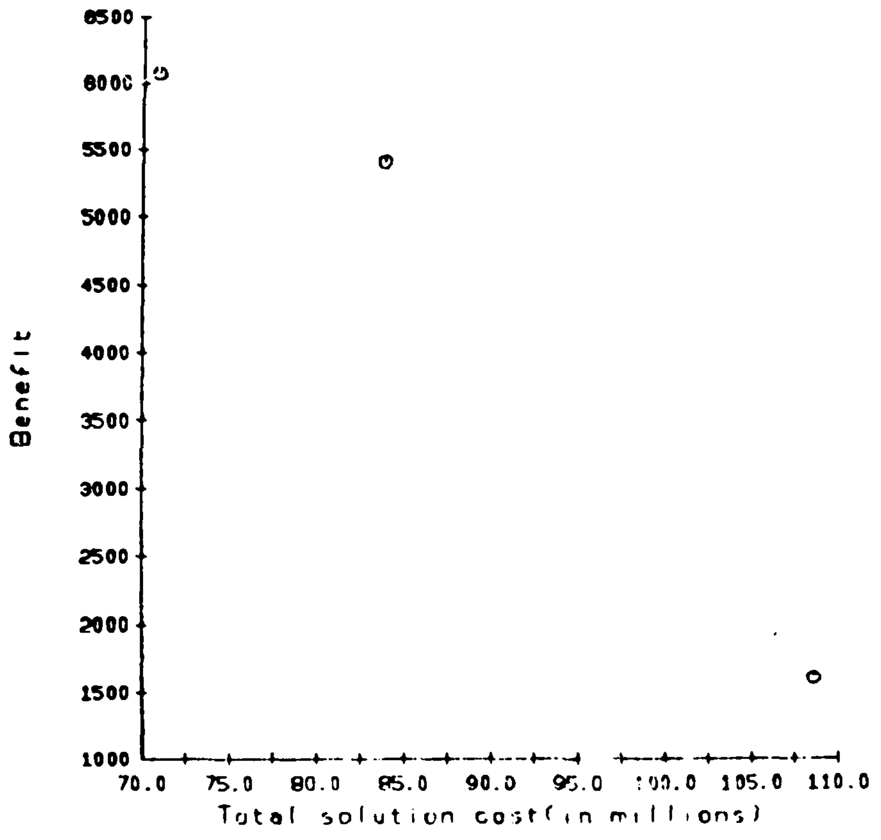


FIGURE 5
SAMPLE GRAPH TYPE 3

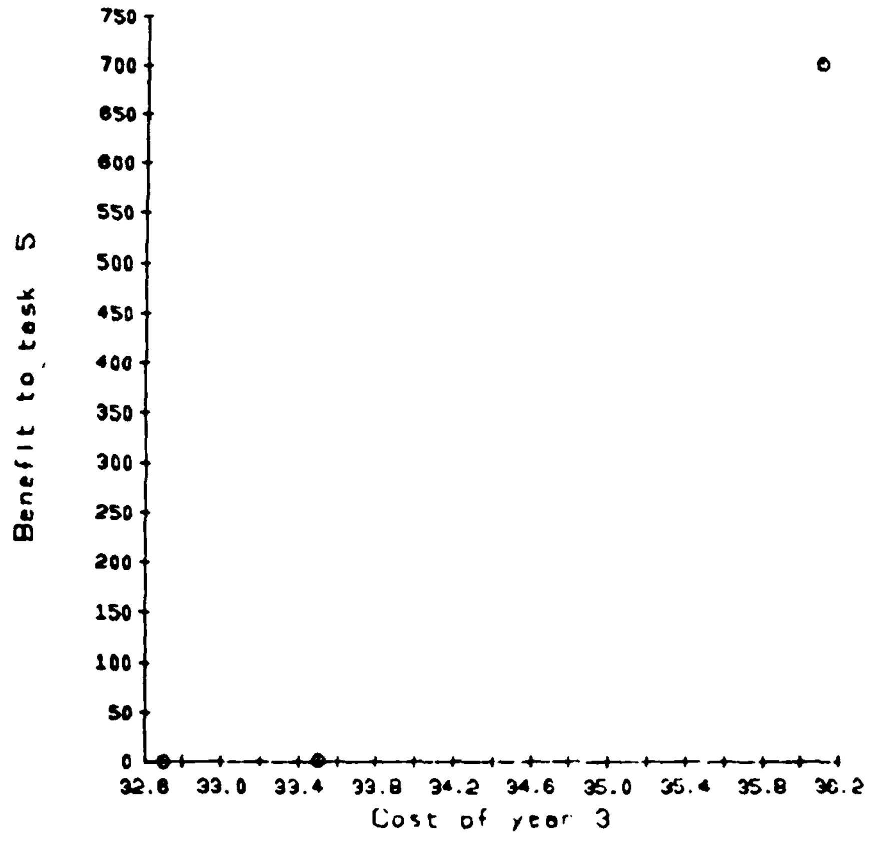
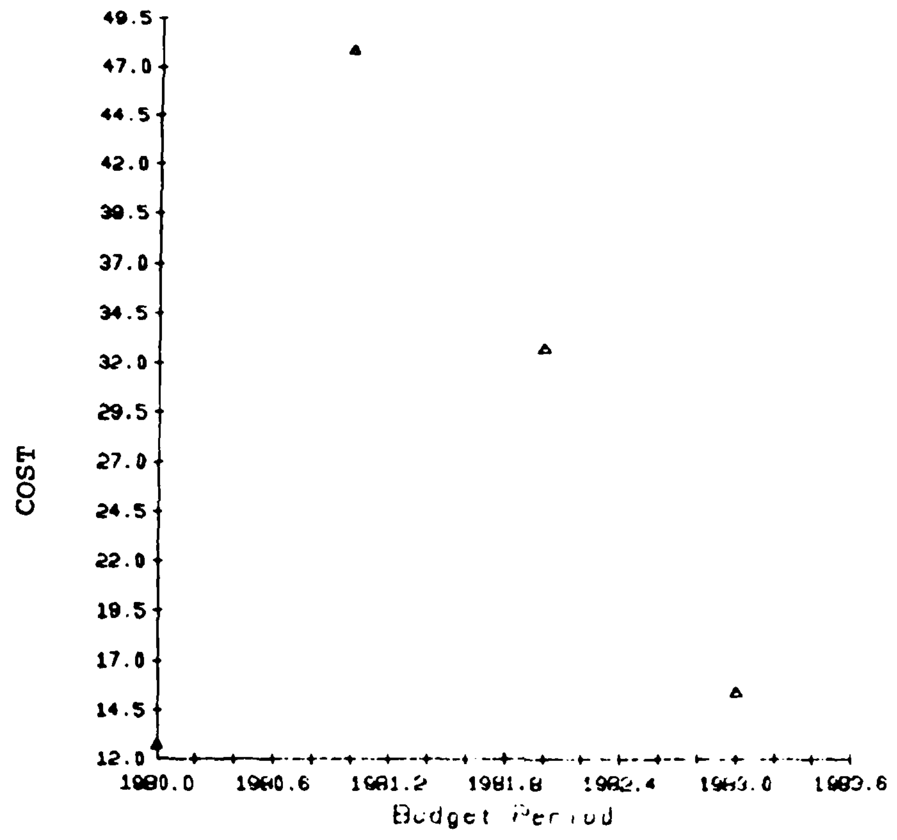


FIGURE 6
SAMPLE GRAPH TYPE 4



For each graph type, you are asked if you want any graphs of that type. If the answer is "yes," for Graph Types 1, 3, and 4, you are asked to input the appropriate benefit measure indices or budget period indices. Note that you cannot use the budget period and task directly; rather, you must use the appropriate index. Thus, if you desire a graph involving 1990 (the third budget period), you must input the index "3", not 1990.

Since Graph Type 2 has no alternatives associated with it, the system will simply ask whether or not you want it. Answer "yes" or "no"; no other inputs are necessary

After each graph is produced, a question mark will flash in the upper lefthand corner of the screen. You may now make a copy of the graph on the Tektronix 4631 photocopier. To inform the subroutine that you are ready to move on to the next graph, type "q" and hit the return key. You should clear the screen before typing "q" to avoid screen clutter. If you do not, the next question will be printed over the graph, which is not automatically removed from the screen.

Graph Types 1, 2, and 3 pose a problem in that each point plotted on these graphs is an output from a different strategy, but no information is embedded in the graph to tell you which point is associated with which strategy. We recommend the following procedure to provide identification. Before displaying the graph, the subroutine will print the data to be graphed. This information occurs in two rows of data. The first entries in each row correspond to the inputs from strategy 1, the second entries in each row correspond to the inputs from strategy 2, and so forth. For example, if you have answered "yes" to the question, "Do you want any graphs of total solution costs summed over all years vs.

benefit of solutions to individual tasks?", you will be asked to key in a task index. Assume you have keyed in index "5," and that three solutions have been saved. The following data will be displayed:

```
    solution cost data              83.90     70.90   108.60
    solution benefit data          700.01      0.70     0.00
```

(see Figure 3). The first two values in each row, 83.90 and 700.01, represent the point (83.90, 700.01) resulting from strategy 1. The second two values represent the point (70.90, 0.70) from strategy 2, and so forth. We recommend that you make a copy of these data when they are flashed on the screen. If you press the "copy" key as soon as the data are presented, the copy can be made before the subroutine is ready to present the graph. Hopefully, this entire problem will be solved by future modifications to the subroutine.

Another problem, which can easily be corrected, is that once you have answered the question, "Do you want to generate more data before constructing graphs?" with a "no," you will exit from subroutine GRAPH, and all data stored in it will be lost. Thus, no data are saved after the graphs have been made.

You may wonder why the points in the various graphs are not connected with lines. The reason is that the quantities being graphed do not change in a linear fashion; rather, values tend to jump suddenly. Furthermore, the parameter changes that will cause a change in the optimal solution are not predictable without extensive sensitivity analysis. Thus, "connecting the dots" in the usual linear fashion is mathematically meaningless.

The last concern is that graphs can be produced only from Tektronix CRT terminals that are attached to the Multics computer system.

III. PROGRAMMER'S GUIDES

A. Programmer's Guide to Program NOW

1. Background

This program acts as an interface between the user, the data base, and the allocation and graphic subroutines. The user inputs the mission area, appropriation, and years under consideration and their budget ceilings. The program then retrieves program elements (PE) numbers and alternatives from the data base and indexes them for input to the allocation routine. The results are passed to the output subroutine and then to the graphics subroutine. When subroutine GRAPH is finished, the program run is essentially finished, and the user may stop or choose one of the other options to generate a new problem.*

The main program is written in PL/1, and some of its significant features are:

- The range of a DO loop is terminated at the first "end" statement, which is equivalent to the FORTRAN "continue" with the appropriate statement number in front.
- The concatenation operation joins two or more strings together into one larger string.
- The four subroutines that are written in PL/1 are all physically contained in the main program. This means that a variable in the main program is known by the same name in the subroutines, as long as it is not declared in that subroutine. This eliminates the need for argument lists in the call statements to these subroutines.

*For details on how to solve problems with this routine, see "User's Guide to Program NOW," Section II.A.

- o PL/1 stores its array elements in row-major order; that is, the rightmost subscript varies most rapidly. This differs from FORTRAN, which uses column-major order. Here, the dimensions of any two-dimensional array in the main program must be reversed from the dimensions of the corresponding array in either of the FORTRAN subroutines, before the values can be passed correctly through the argument list.

The program accesses the data base through the built-in dsl subroutines to retrieve the necessary data. Each dsl call is usually followed by a statement that calls the com_err_ subroutine if the variable code is not equal to zero. This is only the case if something abnormal has occurred during the dsl subroutine, and com_err_ will print out a description of the error that caused the nonzero code.

A new search rule must be added to the user's ordinary search rules so that the system's graphics subroutines can be located when called. The add_search_rules_ (asr) command accomplishes this task and may be done automatically if the command is placed in the user's start-up.ec.

2. Program Description

This program is described sequentially. Line numbers have been added for convenient reference and are keyed to the computer code listed in Section IV.A of this volume.

Program Description

Line No.	
1	Procedure statement - entry point to program
2-9	Declaration of all external subroutines
10-16	Declaration and definition of a block for the quit (interrupt) condition. If the break key is passed while the program is running, control will be transferred to this block, which ensures that the data base is properly closed before returning to command level
17-18	Declaration of integer variables and arrays
19	Declaration of binary-based floating point data, which corresponds to the FORTRAN "real" data
20	Declaration of the cost data as fixed decimal to correspond with the numbers in the data base
21-29	Declaration of all character data
30-35	Initialization of some character data to the different parts of a dsl subroutine call's selection expression
36	Initialization of the variable path to the path name of the proper data base
37	delta is set to a small positive number
38	rseed is set to any random seed
39	Initialization of five different variables to zero
40-41	Person_id is asked for and read in
42-43	User is asked to input rd or sc for data base, and the response is read into the data base

44-48	The data base is opened. If an error occurs, com_err_ is called and processing is halted
49-50	Scope is set on the "protect" submodel, and com_err_ is called if an error occurred
51-58	The first mission area that the user has access to is retrieved, and the usual precautions are taken in case of an error
59	A mission area access heading is printed out
64-65	The user is asked if he wants official data, and yes(y) or no(n) is read into dc
66-67	The user is asked to input the mission area, and the response is read into subma
68-73	If the user entered "none" for mission area, then the data base is closed and processing stopped
74-75	If the user wants official data then ma is set equal to "off"; otherwise it is set to the mission area
76-78	The name of the submodel that the user will be accessing to obtain PE numbers and cost data is retrieved, and com_err_ is called if an error occurs
79-85	Scope is deleted, the data base is closed, and com_err_ is called if either code does not equal zero
86-87	Ask for appropriation and read into appro
88-89	Ask for number of years to be considered, and read into nper
90-93	Ask for year and budget ceiling for each period, and read into nyr( ) and b( )
94-104	This section opens the proper submodel and sets scope on "char" and "relation." Char is the part of the data base that contains PE numbers and their different funding alternatives. Relation is a variable that represents the part of the data base that contains the appropriate cost data

	This section is skipped if dec equals five. This can happen only if the user selected option five after the first allocation was completed. If this is the case, then these instructions are unnecessary and so can be omitted
105	The number of groups (PE numbers) is initialized to zero
106	exp8 is set to a character string that forms the selection expression of the data base call that will retrieve PE numbers given the mission area and appropriation
107	The first PE number is retrieved
108-117	This block is activated if the preceding retrieval resulted in an error. An error message and three user options are printed out, and control is transferred as desired
118-122	This loop retrieves all remaining PE numbers and places them in penum( )
123	The number of variables (a PE and an alternative) is initialized to zero
124	exp7 is set to a character string that represents the selection expression of a data base call that will retrieve the alternatives and cost data for each PE
125	A heading is printed out for any zero cost alternatives that may be found
126-175	This loop retrieves all the alternatives and the cost data for each for a given PE. If any alternative has cost equal to zero for all the years under consideration, then it will not be indexed as a variable. If all the alternatives under a PE have zero cost, then that PE will be eliminated and ngr reduced by one
127	If i equals the reduced number of groups, then go to the next i

130	The i^{th} PE number is obtained from penum( )
131	The first alternative and its associated cost data are retrieved
136	Subroutine DCOST is called
137-141	The current alternative is checked for zero cost. If yes, then the PE and alternative are printed out and flag1 and jj are set to 1. Flag1 = 1 signifies that the first alternative under this PE has zero cost. jj is changed from 2 to 1 because k, which indexes the remaining alternatives under this PE, is started at jj and in this case will be the first, not the second, variable
142-148	If the first alternative was a valid one, however, then this section is processed instead of the previous one
143	The number of variables is incremented
144	Subroutine COST is called to store the cost data of the current variable
145-147	The variable is indexed into the group number and the alternative number under that group and the funding level is stored in the progalt array
149-165	This loop retrieves the remaining alternatives and their cost data under the current PE. It checks for zero cost and indexes in the same manner as that for the first alternative
166-174	This section is processed only if flag1 = 1 and flag2 = 0, which can happen only if no nonzero cost alternatives occurred under the current PE. In this case, the penum array is adjusted to eliminate that PE number, the number of groups is reduced by one, and i is decremented so that a number will not be skipped on the next pass through the loop
176-178	The allocate, output, and graph subroutines are called in order. Allocate and graph are referenced by <filename>$<program name>, since they are located in a storage region outside of the main program

179-182	The user options are printed and the response read into dec
183	If the user asks for a different appropriation, then control is transferred to statement 12
184-193	This section is for a different data base or different mission area options. Scope is deleted, the data base is closed, and comm_err_ is called in case of error. Then control is transferred to the appropriate place
194-204	This section is for the different budget ceilings option. The number of years with new budgets is asked for and read into nchange. Then each year and its new budget level are read in and control transferred as appropriate
205-211	This section is for the option to choose different years of cost data. All budget ceilings are zeroed out, and control is transferred back to where the years and budget ceilings are input
212-215	This section can be reached only if the user elects to stop. Scope is deleted, the data base is closed, and processing is halted

Description of Subroutines DCOST and COST

DCOST:	This subroutine checks all alternatives for zero funding. It totals the cost of the current alternative over the years under consideration and returns that value in asum. When dcost is finished processing, asum is immediately checked for a zero value.
COST:	Once an alternative has been determined to be a legitimate variable, subroutine cost is called to store its cost data. The main program uses the array cst, with cst(i,j) the cost of variable j in budget period i. Note that these dimensions are reversed in the corresponding c array in the two fortran subroutines.

output:

2-8	Declarations and initializations
9-11	Print instructions and read in a dummy character
12	Set the entire sum array equal to zero
13-17	Since six alternatives will be printed per page, this block determines how many pages will be required and how many alternatives will be left for the last page.
18	Skip two lines of output
19	Concatenate a data base variable to represent the selection expression necessary to retrieve the cost data for each variable
20-50	This loop prints a page of output each time through
21-23	Print page heading
24-25	Set upper limit on inner loop to 6, unless this is the last page to be printed
26-45	This loop prints an alternative and its associated data each time through

27-29	Determine the PE number and funding alternative for the current variable
30-35	Retrieve the PE title for the current variable, call com_err_, and exit the loop if an error occurs
36-37	Retrieve the cost data for the current variable, and exit the loop if an error occurs
38	Call subroutine sumcost to maintain a running total of the cost of the solution for each budget period
39-45	Print out the current variable and its cost and benefit data, and repeat loop until ii > index1
46	If all output is finished, print the cost total for each budget period
47-48	Print another heading, and skip two lines
49	Read in a dummy character
50	End the page-printing loop

3. Definition of Variables

nvar:	The number of variables. A variable is a specific funding alternative with nonzero cost under a given PE number
ngr:	The number of groups or PEs under a given mission area and appropriation with at least one nonzero funding alternative
pe:	PE number under consideration
penum ( ):	Contains all legitimate (nonzero funding) PE numbers
appro:	Appropriation
pealt:	Funding alternative under consideration
progalt (n):	Funding alternative of variable n
ma:	Equals "off" for official data; otherwise equals mission area
subma:	Mission areas, 000 to 900; used to retrieve PE numbers
nper:	Number of budget periods that will be considered for the problem
nyr (n):	n^{th} year, two digits, 79-85
ny:	Year index number, = nyr(y) - 78
budget:	Budget ceiling for a given year
b(n)	Budget ceiling for year n
yr1-yr7:	Cost data for a given variable
cst (i,j):	Cost of variable j in budget period i
sum(n):	Total cost of solution in period n

delta:	Small positive number, to check for zero cost alternatives
asum:	The total cost of an alternative, summed over all periods under consideration
nchange:	User option, number of years with a new budget ceiling
init:	Passes to subroutine GRAPH the number of times that it has been called minus 1
dummy:	A 1-character dummy variable used to temporarily halt the output so that the user may reset page if a CRT is being used
exp1 exp8:	Character strings that contain parts of the various data base selection expressions
sub- model:	The submodel (subset) of the data base that will be accessed
data base	The path name of the proper submodel
dbi, dbi1:	Data base index numbers, returned from a dsl_$open call
code:	Returned by all dsl subroutines; code equals zero if the data base call was performed successfully; otherwise it equals a particular number, depending on the exact error that disabled the subroutine
db:	Data base, rd or sc as user desires
rela- tion:	The specific area in the data base that contains the appropriate cost data
ilev (n):	Index number of variable n under its PE number

igr(n):	Index number of the PE to which variable n belongs
js(n):	Index number of n^{th} solution variable
jl:	Number of variables in the solution
zw(i,j):	Benefit to task i from variable j
ng:	Number of goals (tasks)
xl(),yl(), y2():	Arrays that preserve the values of three arrays of subroutine GRAPH in case it is called more than once

B. Programmer's Guide to Subroutine GRAPH

1. Program Description

This section describes in detail the workings of subroutine GRAPH. You* should read it with a subroutine listing in hand (see Section IV.B). An alphabetical listing of variable definitions is given in Section III.B.

Before starting into subroutine GRAPH, we should discuss the parameter j, which is passed to GRAPH in the subroutine call statement. The parameter represents the number of sets of resource allocation output data you have saved to be graphed. You must set j equal to zero in the calling program before calling GRAPH.

The variables c, uper, js, ngr, nvar, ng, nyr, zw, jl, x, y, and yr are listed as subroutine arguments because the calling program is written in PL/1. A PL/1 program may not share variables with a FORTRAN subroutine.

The first DO loop, DO 2, and the statement preceding it

$$x(j+1)=0.0,$$

initialize x(j+1) and yr(L,j+1), L=1,...,nper. The DO 5 loop computes the total solution cost x(j+1) and the cost per-budget period of the solution yr(L,j+1)

$$x(j+1)= \sum_{i=1}^{jl} \sum_{L=1}^{nper} c(js(i),L)$$

$$yr(L,j+1)= \sum_{i=1}^{jl} c(js(i),L) \quad ; \; L=1,\ldots,nper,$$

where L indexes budget periods and i indexes the selected program alternatives. The total solution cost is printed out at this time.

*You in this section refers to you, the programmer. However, at times you will be taking on the role of user as well.

The next DO loop, DO 10, computes the contributions of the solution to the goals with respect to which benefits are measured (also called tasks), y(m,j+1), for each task m. Thus

$$y(m,j+1) = \sum_{i=1}^{j1} zw(js(i),m) \quad , \; m=1,\dots,ng.$$

The total benefits for these tasks are also printed out.

At this point, you have the total solution cost and the total solution benefit as follows:

```
The total cost of the solution is      106.60
      contribution to task  1 =     11.00
      contribution to task  2 =     33.00
      contribution to task  3 =     31.00
      contribution to task  4 =   1520.00
      contribution to task  5 =      0.00
      contribution to task  6 =      0.00
      contribution to task  7 =      2.50
      contribution to task  8 =      1.85
Do you want the output from this strategy saved for graphs. Please answer yes
\cor no.
yes
do you want to generate more data before constructing graphs?
no
Do you want any graphs of total solution costs summed over all years vs  contri
\cbution of solution to individual benefit measures
yes
Key in index for the benefit measure.
4
```

You are then asked whether you want to save the data associated with this solution for incorporation into graphs. If you answer "yes," j is incremented by 1,

$$j \longleftarrow j+1.$$

Since the value j+1 was used in the DO 5 and DO 10 loops, incrementing j has the effect of saving the results of these loops. Also at this time, the sum of the benefits of the solution, y(ngl,j), is computed in the DO 15 loop,

$$y(ngl,j) = \sum_{m=1}^{ng} y(m,j).$$

You are then asked if you want to generate more data using additional investment strategies before constructing graphs. This questi~ is also asked if you answered "no" to the preceding question. (However, if you did answer "no," j will not be incremented, and the next time GRAPH is called, the current stored values will be lost.) If you want to generate more data, control is passed back to the calling program. If you do not, you are taken to the portion of the subroutine that generates the graphical output. We will proceed under the assumption that you answered "no."

The first graph type (see Figure 3 for sample)* that can be generated is the contribution of the solution to individual tasks y(i,k), m+1,...,ng versus total solution costs summed over all years, x(k), for each solution k, k=1,...,j. Thus, a graph consists of a set of points {(x(k),y(i,k)), k=1,...,j}. You are asked if you want any graphs of this type. If you answer "yes," you are then asked to key in the index i for your desired task. The index is an integer from the set {1,2,...,ng} referencing one of the tasks. If you answer "no," control jumps to the "101 continue" statement for entry into next graph type. We will proceed under the assumption that you answered "yes."

*For a summary of all graph types offered, see Figure 2.

Once you have selected the benefit measure index i, the DO 27 loop stores the value of y(i,k) in the single subscripted variable z(k), k=1,...,j. This is necessary, since the "xyplot" statements used to construct the graph will not accept a double subscripted variable.

The total solution cost x(L) is written out for each solution L, L=1,...,k, and the contribution of the solution to task i,y(i,L) is written out for each solution L, L=1, ...,j. You should make a copy of this information when it appears on the screen, since you will have no other way of knowing which point on the graph is associated with which strategy.

The encode and associated statements that now occur are used to concentrate the task index i to the rest of the graph's title, stored in zz, and the rest of the y-axis title, stored in ben. If you are unfamiliar with encode statements, you should consult Multics system documentation before tampering with encode statements, variables used in encode statements, or format statements associated with encode statements. The encode statement is discussed on pages 5 through 12 of the *Multics FORTRAN Guide*.

The "call xyplot" statements that now occur call system subroutines that construct and display the graph. A complete discussion of the xyplot statements is contained in the document *XYPLOT*, which was prepared by, and is available from, Mr. Robert G. Finney of the Air Force Data Services Center.

Once the graph has been displayed, a question mark will appear in the upper lefthand corner of the screen. This indicates that you may now make a copy of the graph on the Tektronix 4631 photocopier. Once you have made a copy of the displayed graph, clear the screen to avoid the clutter that arises from overprinting on the graph (the graph will not be removed automatically). Then inform the system that you are ready to move on to another graph by typing "q". Upon receipt of the "q" signal, the subroutine returns to the question, "Do you want any graphs of this type?" If you do, the preceding sequence is repeated. If you do not, the subroutine goes to the "101 continue" statement to initiate proceedings on the next type of graph.

The next statement after the "101 continue" statement asks whether you would like a graph of total benefit versus total cost for each alternative (see Figure 4 for sample). If you do not, control is shifted to the "206 continue" statement before the next set of graphs. If you do, the subroutine proceeds to construct this graph. The DO 110 loop stores the sum of the benefits of the kth strategy y(ngl,k) in the single subscripted variable z(k) because "xyplot" statements will not accept double subscripted variables as inputs.

Next the system displays the x and y coordinates of the points to be graphed, (x(L),L=1,j) and (y(ngi,L), L=1,...,j). As we have mentioned previously, you should make a copy of this information on the Tektronix 4631 copier. The "call xyplot" statements create and display the graph of the set of points {(x(L),y(ngl,L), L=1,...,j}.

The next graph type shows individual task benefits for each strategy versus individual budget periods (see Figure 5 for sample). If you want a graph of this type, key in the budget period you desire and the benefit measure index. The DO 214 loop stores the double subscripted variables y(n,k) and yr(i,k) in the single subscripted variables z(k) and v(k), respectively. The arrays of numbers to be graphed {yr(i,L),L=1,j} and {y(n,L), L=1,j} are then printed out.

The encode statements set up the title and x-axis and y-axis labels. The xyplot statements construct and display the graph.

After you have viewed the graph and signaled the system to continue with the program, control is returned to the "201 continue" statement. You are then asked if you want any more graphs of this type. If so, the sequence repeated. If not, control is transferred to the "301 continue" statement for the final set of graphs.

The last graph type shows the cost of an individual strategy, or solution, for each budget period (see Figure 6 for sample). Thus, the set of points graphed is {(yr(i,n),nyr(i), i=1,...,nper}. This graph is automatically produced for each n, n=1,...,j if you want any graphs of this type.

When the "401 continue" statement is executed, the graphing is completed. If you set j equal to zero, you will reinitialize subroutine GRAPH. Data currently stored in subscripted variables will be overwritten the next time subtoutine GRAPH is called by program NOW.

2. Definitions of Variables Used in Subroutine GRAPH

ben	Character-string variable used in constructing the legend for the y-axis on several of the graphs
c(js(i),l)	Cost of the js(i)th program alternative during the lth budget period
cost	Character-string variable used in constructing the legend for the x-axis on one of the graphs
iz	Used in an xyplot$build statement to indicate the type of symbol to be used to designate plotted points

j	Denotes the number of different solutions to the resource allocation problem that have been stored in subroutine GRAPH; each time the user answers the question "Do you want the output from this strategy saved for graphs?" with a "yes," j is incremented by 1, and the solution is saved
j1	Number of selected PE alternatives
ng	Number of goals or tasks toward which resources are allocated by the resource allocation algorithm
ng1	Used in several places as a temporary storage location for the value ng+1
nper	Number of budget periods over which program costs have been computed; budget periods would typically be in fiscal years
nyr(i)	The ith budget period; for example, if the second budget period is 1990, the nyr(2)=1990
title	Character variable used to store the title of each graph; previous content stored in title must be deleted before new content is inserted
tle	Character variable used in building up the legend for the y-axis of one of the graphs
tpe	Character variable used in building up the legend for the y-axis of one of the graphs
tse	(same as above)
v(k)	Used to store yr(i,k) for each k; necessary because arguments called by xyplot$build must be variable with a single subscript, so a graph using the values stored in yr(i,k) cannot be built using yr(i,k) as an argument
x(j)	Stores the total cost of the solution corresponding to the jth strategy that the user saves to be graphed

xlabel	Character variable that is used in storing the legend for the x-axis of each of the graphs
y(m,j)	Stores the benefit of the jth strategy or solution to the mth task
y(ngl,j)	Stores the sum of the benefits of the jth strategy
ylabel	Character variable that is used in storing the legend for the y-axis of each of the graphs
yr(l,j)	Stores the total cost of the jth strategy or solution in the lth budget period
z(l)	Used to store the value of y(i,l) for each l; necessary because arguments called by xyplot$ build must be variables with a single subscript, so a graph using the values of y(i,l) cannot be built using y(i,l) as an argument
zw(js(i),m)	Stores the value of the contribution of the js(i)th program alternative to the mth benefit measure
zz	Character variable used to store the title of each of the graphs

IV. COMPUTER CODES

A. Program NOW

```
pr now.pl1

                        now.pl1   11/05/79  1224.1 est Mon


 now: proc options(main);
 dcl (ram$allocate,gph$graph) entry options(variable);
 dcl dsl_$open entry options(variable);
 dcl dsl_$close entry options(variable);
 dcl dsl_$dl_scope_all entry options(variable);
 dcl dsl_$dl_scope entry options(variable);
 dcl dsl_$set_scope entry options(variable);
 dcl dsl_$retrieve entry options(variable);
 dcl com_err_ entry options(variable);
 dcl quit condition;
 on quit
   begin;
   call dsl_$close(dbi1,code);
   call dsl_$close(dbi,code);
   stop;
   end;
 dcl(dbi,dbi1,dec,code,ilev(350),igr(350),nyr(10),sum(7),ng,
 perm,ny,j1,js(80),nvar,ngr,nper,flag1,flag2) fixed binary;
 dcl(y1(10,25),x1(10),cst(10,350),b(10),rseed,delta,budget,zw(4,500)) binary floa
 dcl(yr1,yr2,yr3,yr4,yr5,yr6,yr7,asum) fixed decimal(8,0);
 dcl(pealt,progalt(350),db) char(2),dc char(1),
 (ma,answer) char(3),
 appro char(4), (pe,penum(80)) char(6),
 dp char(9), (submodel,exp1) char(10),
 relation char(8), subma char(3),
 exp2 char(19), exp5 char(20),path char(25),
 (name,person_id) char(15), acct char(32),
 database char(35), exp3 char(46),exp8 char(103),
 exp4 char(130), exp7 char(148);
 exp1="-range (p ";
 exp2=") (c char) -select ";
 exp3="p.pe -where (((c.ma_x00 = .V.) & (c.pe = p.pe)";
 exp5=") & (p.appro = .V.))";
 exp4=") -select p.dp p.yr1_dol p.yr2_dol p.yr3_dol p.yr4_dol p.yr5_dol p.yr6_dol
 "p.yr7_dol-where ((p.pe=.V.) & (p.appro = .V.))";
 patn=">udd>RDIS>rdis_lib>RD_DB>";
 delta=.000001;
 rseed=74852399.0;
 init,dbi,dbi1,dec,ngr1=0;
 put skip list("Enter your Multics person_id");
 get list(person_id);
 14: put skip list("What database? rd or sc?");
 get list(db);
 15: call dsl_$open(">udd>RDIS>rdis_lib>RD_DB>protect.dsm",dbi1,1,code);
 if(code^=0) then
   do; call com_err_(code,"opening protect");
   stop;
   end;
```

```
  call dsl_$set_scope(dbil,"protect",l,0,30,code);
  if(code^=0) then call com_err_(code,"setting scope on protect");
  call dsl_$retrieve(dbil,"-range(p protect)-select p.ma p.permission -where(p.
  person_id,ma,perm,code);
  if(code^=0) then
    do; call com_err_(code,"retrieving ma's");
    call dsl_$close(dbil,code);
    if(code^=0) then call com_err_(code,"closing database");
    stop;
    end;
   put skip list("You have access to the following  mission areas:");
   do while(code=0);
    put skip list(ma);
    call dsl_$retrieve(dbil,"-another",person_id,ma,perm,code);
    end;
  put skip(2) list("off data?");
  get list(dc);
  l1: put skip list("Which mission area? if none, type 'none'.");
  get list(subma);
 if(subma="non") then
   do; call dsl_$dl_scope_all(dbil,code);
   call dsl_$close(dbil,code);
   if(code^=0)then call com_err_(code,"closing database");
   stop;
   end;
  if(dc="y") then ma="off";
    else ma=subma;
    call dsl_$retrieve(dbil,"-range(p protect)-select p.submodel-where((p.ma=.V
    ma,person_id,submodel,code);
    if(code^=0) then call com_err_(code,"Choosing ma's");
   l2: call dsl_$dl_scope_all(dbil,code);
  if(code^=0) then call com_err_(code,"deleting scope on protect");
  call dsl_$close(dbil,code);
  if(code^=0) then
    do; call com_err_(code,"closing protect");
    stop;
    end;
   put skip list("What appropriation?");
  get list(appro);
  l7: put skip list("How many years of cost would you like to consider?");
  get list(nper);
   do i=1 to nper;
    put skip list("Enter year, budget");
    get list(nyr(i),b(i));
    end;
  if(dec^=5) then
  do; database=path||submodel;
  call dsl_$open(database,dbi,l,code);
  if(code^=0) then
      do; call com_err_(code,"opening database");
      stop;
      end;
    relation=db||"rp_"||ma;
    call dsl_$set_scope(dbi,"char",1,0,relation,1,0,code);
    if(code^=0) then call com_err_(code,"setting scope on database");
    end;
  ngr=0;
  exp8=exp1||relation||exp2||exp3||exp5;
  call dsl_$retrieve(dbi,exp8,subma,appro,pe,code);
  if (code^=0) then
    do; call com_err_(code,"no pe");
    put skip edit("No pe was found for mission area",ma," with appropriation",appro
    "1=different mission area and appropriation","2=different appropriation","3=sto
    col(1),a,col(1),a);
    get list(dec);
    if(dec=1) then go to 11;
    if(dec=2) then go to 12;
    else go to 18;
    end;
    do while(code=0);
    ngr=ngr+1;
    penum(ngr)=pe;
    call dsl_$retrieve(dbi,"-another",subma,appro,pe,code);
    end;
  nvar=0;
  exp7=exp1||relation||exp4;
  put skip list("The following pe's and alternatives have zero funding for the spec
    do i=1 to ngr;
    if(i=ngr1) then go to 16;
    jj=2;
    flag1,flag2=0;
    pe=penum(i);
    call dsl_$retrieve(dbi,exp7,pe,appro,pealt,yr1,yr2,yr3,yr4,yr5,yr6,yr7,code);
    if(code^=0) then
      do; call com_err_(code,"retrieving cost data");
      go to 13;
      end;
    call dcost;
    if(asum<=delta) then
     do; put skip list(pe,pealt);
      jj=1;
     flag1=1;
     end;
    else
      do; nvar=nvar+1;
      call cost;
      igr(nvar)=i;
      ilev(nvar)=1;
      progalt(nvar)=pealt;
      end;
      do k=jj by 1;
      call dsl_$retrieve(dbi,"-another",pe,appro,pealt,yr1,yr2,yr3,yr4,yr5,yr6,yr7,
    if(code^=0)then go to 112;
      call dcost;
```

```
if(dec=5) then
  do; do i=1 to nper;
       nyr(i)=0;
       b(i)=0.0;
       end;
  go to 17;
  end;
18: call dsl_$dl_scope_all(dbi,code);
call dsl_$close(dbi,code);
if(code^=0) then call com_err_(code,"closing database");
stop;

dcost: proc;
asum=0.0;
  do ii=1 to nper;
  ny=nyr(ii)-78;
  if(ny=1) then asum=asum+yr1;
  else if(ny=2) then asum=asum+yr2;
  else if(ny=3) then asum=asum+yr3;
  else if(ny=4) then asum=asum+yr4;
  else if(ny=5) then asum=asum+yr5;
  else if(ny=6) then asum=asum+yr6;
  else asum=asum+yr7;
  end;
end dcost;

cost: proc;
  do ii=1 to nper;
  ny=nyr(ii)-78;
  if(ny=1) then cst(ii,nvar)=yr1;
  else if(ny=2) then cst(ii,nvar)=yr2;
  else if(ny=3) then cst(ii,nvar)=yr3;
  else if(ny=4) then cst(ii,nvar)=yr4;
  else if(ny=5) then cst(ii,nvar)=yr5;
  else if(ny=6) then cst(ii,nvar)=yr6;
  else cst(ii,nvar)=yr7;
  end;
end cost;

 output: proc;
dcl exp3 char(10), exp4 char(153), title char(25),
rank char(4), comment char(30), exp7 char(171),
alt char(2), dummy char(1),exp6 char(10);
expo="-range (d ";
exp3="-range (p ";
exp4=")-select p.yr1_dol p.yr2_dol p.yr3_dol p.yr4_dol p.yr5_dol p.yr6_dol p.y
"p.comment1-where (((p.pe = .V.) & (p.appro = .V.)) & (p.dp = .V.))";
put skip edit("1.Type in any character, reset page, and carriage return.",
"After each page has finished printing, repeat step 1.")(a,col(1),a);
get list(dummy);
sum=0;
index=(j1-1)/6+1;
```

```
        if(asum<=delta) then
          do;  k=k-1;
          put skip list(pe,pealt);
          end;
        else
          do; nvar=nvar+1;
          igr(nvar)=i;
          ilev(nvar)=k;
          progalt(nvar)=pealt;
          call cost;
          flag2=1;
          end;
      end;
  112: if(flag1=1 & flag2=0) then
          do;
             do j=1 to ngr-1;
             penum(j)=penum(j+1);
             end;
          ngr1=ngr;
          ngr=ngr-1;
          i=i-1;
          end;
    13:end;
    16: call ram$allocate(cst,nper,ngr,nvar,ilev,igr,b,rseed,nyr,sum,zw,dec,js,ng,jl
   call output;
   call gph$graph(init,cst,nper,js,ngr,nvar,ng,nyr,zw,jl,xl,yl);
   put skip edit("Select one of the following:","1=different database","2=different
   "3=different appropriation","4=different budgets for years currently considered",
   "5=different years of cost data","6=stop")(a,col(1),a,col(1),a,col(1),a,col(1),a,
   get list(dec);
   if(dec=3) then go to 12;
   if(dec=1 | dec=2)then
     do; call dsl_$dl_scope_all(dbi,code);
     call dsl_$close(dbi,code);
     if(code^=0) then
       do; call com_err_(code,"closing database");
       stop;
       end;
     if(dec=1) then go to 14;
      else if (dec=2) then go to 15;
     end;
    if(dec=4) then
     do; put skip list("Enter number of years with new budgets");
     get list(nchange);
       do i=1 to nchange;
       put skip list("Enter year, new budget");
       get list(ny,budget);
       n=ny-78;
       b(n)=budget;
       end;
     go to 16;
     end;
  irem=mod(j1,6);
  indl=6;
  if(irem^=0) then indl=irem;
  npe=0;
  put skip(2);
  exp7=exp3||relation||exp4;
20 do i=1 to index;
    put skip edit("***SECRET***","Page",i)(col(40),a,skip,col(75),a,f(2));
    put skip edit("Allocation report of mission area ",subma," using ",ma," data")(
    put skip edit("Cost in thousands of dollars")(col(30),a);
    indexl=6;
    if(i=index) then indexl=indl;
      do ii=1 to indexl;
      npe=npe+1;
      pe=penum(igr(js(npe)));
      alt=progalt(js(npe));
      call dsl_$retrieve(dbi,"-range (c char) -select c.pe_title-where ((c.pe=.V.)
      pe,subma,title,code);
      if(code^=0) then
        do; call com_err_(code,"returning pe");
        go to 113;
        end;
      call dsl_$retrieve(dbi,exp7,pe,appro,alt,yr1,yr2,yr3,yr4,yr5,yr6,yr7,comment,
      if(code^=0) then go to 113;
      call sumcost;
      put skip(2) edit("PE","PE TITLE","APPRO     DP  ","COMMENTS")(x(5),a,x(12),a,
40    put skip edit(pe,title,appro,alt,comment)(x(3),a(6),x(2),a(25),x(2),a(4),x(5)
      put skip(2) edit("1979","1980","1981","1982","1983","1984","1985","TSK1","TSK
      (x(13),a,x(5),a,x(6),a,x(6),a,x(6),a,x(6),a,x(6),a,x(4),a,x(4),a,x(4),a,x(4),
      put skip edit("COST",yr1,yr2,yr3,yr4,yr5,yr6,yr7,(zw(jj,js(npe))do jj=1 to ng
      put skip(2);
      113: end;
    if(i=index) then put skip edit("Total",(sum(l)do l=1 to 7))(x(1),a,x(3),(7)f(10
    put skip edit("***SECRET***")(skip(3),x(40),a);
    put skip(2);
    get list(dummy);
    end;

  sumcost: proc;
  sum(1)=sum(1)+yr1;
  sum(2)=sum(2)+yr2;
  sum(3)=sum(3)+yr3;
  sum(4)=sum(4)+yr4;
  sum(5)=sum(5)+yr5;
  sum(6)=sum(6)+yr6;
  sum(7)=sum(7)+yr7;
  end sumcost;

  end output;

  end;

r 1226 1.585 0.736 55
```

B. Subroutine GRAPH

```
      subroutine graph(j)
      common/comm1/ sl(10),c(200,10),nper,jflag,slack(10)
      common/comm2/ js(50),jl
      common/comm4/ ign(50),inc(200),cand(50),ngr,nvar,lk,ilev(200),ach(3)
      common/comm10/ ng,nyr(10),sum(7),zw(50,25)
      dimension y(25,10),x(10), benefit(25), a(200),yr(25,10)
      dimension z(25),v(25),nv(10)
      external xyplot$init(descriptors),xyplot$build(descriptors)
      external xyplot$plot(descriptors),xyplot$reset_defaults(descriptors)
      character title*48,xlabel*70,ylabel*20,curve_name*40
      character ben*25,cost*22,til*26,tpe*27,tse*23,zz*41,tle*22
*
*     j is number of sets of data to be graphed.
*     ngr is the total number of program alternatives.
*     nper is the number of budget periods.
*     ng is the number of goals.
*     jl is the number of selected pe alternatives.
*     x(j) is the total cost of the solution for strategy j.
*     benefit(m) is the benefit of the solution to task m.
*     yr(i,j) is the cost if the jth solution in the ith year.
*     y(i,j) is benefit of jth strategy to ith goal.
*     sum(i) is the cumulative cost of current strategy for budget period i, i=1,7.
*     z(i,j) is benefit to jth goal of ith variable, i=1,nvar ; j=1,ng.
*
      x(j+1)=0.0
      do 2 l=1,nper
      yr(l,j+1)=0.0
2     continue
*
*     compute total and yearly costs for solution.
      do 5 l=1,nper
      do 5 i=1,jl
      x(j+1)=x(j+1)+c(js(i),l)
      yr(l,j+1)=yr(l,j+1)+c(js(i),l)
5     continue
*
      write(6,6) x(j+1)
6     format(" ","The total cost of the solution is",f12.2)
*
*     compute task benefits of solution.
      do 10 m=1,ng
      y(m,j+1)=0.0
      do 8 i=1,jl
8     y(m,j+1)=y(m,j+1)+zw(js(i),m)
10    continue
*
      do 11 m=1,ng
      write(6,12)  m,y(m,j+1)
11    continue
12    format (" ",t2,  " benefit to task",i3," =",f10.2 )
*
      print ,     "do you want the output from this strategy
                   saved for graphs?  Please answer yes or no."
      read (5,220) answer
220   format(a3)
      if(answer.eq. "no") go to 1000
*
      j=j+1
      ng1=ng+1
      y(ng1,j)=0.0
*
```

```
*     compute sum of benefits for this strategy.
      do 14 i=1,nq
      y(nq1,j)=y(nq1,j)+y(i,j)
14    continue
*
      print ,"do you want to generate more data before constructing graphs?"
      read(5,220) answer
      if(answer .eq. "yes") go to  1000
*
      nq1=nq+1
*
1      continue
       print ,"'o you want any graphs of total solution costs summed over
              all years vs. benefit of solution to individual tasks?"
      read(5,20) answer
 2    format(a3)
      if(answer .eq. "no")no to 1 1
      print ,"key in index for the task."
      read (5,) i
 *
 *
      do 27 l=1,j
      z(l)=y(i,l)
 27   continue
 *
      write(6 ,30) (x(l),l=1,j)
 3    format(" ","solution cost data",30(f12.2,2x))
      write(6 ,31) (y(i,l),l=1,j)
 31   format(" ","solution benefit data",30(f12.2,2x))
      zz="TOTAL SOLUTION COST vs. BENEFIT FOR TASK"
      encode(title,35) zz,i
 35   format(a40,i3)
      ben="Benefit to task"
      encode(ylabel,32) ben,i
 3    format(a15,i3)
      xlabel="Total solution cost(in millions)"
      call xyplot$init(2,title,xlabel,ylabel,ll)
      call xyplot$reset_defaults(0,0,0,"secret",1," ",0,10.0," ",ll)
      call xyplot$build(j,x,z,01," ",1,ll)
      call xyplot$plot(ll)
 1 0  continue
      no to 19
 *
 1 1  continue
 *
      print ,"'o you want a graph of total benefit vs. total cost for each strategy?"
      read(5,20) answer
      if( nswer .eq. "no") go to 206
 *
      do 110 k=1,j
      z(k)=y(nq1,k)
 110  continue
 *
      write(6,30) (x(l),l=1,j)
      write(6,31)  (y(nq1,l),l=1,j)
      title="TOTAL SOLUTION COST vs. TOTAL BENEFIT"
      xlabel="'otal solution cost(in millions)"
      ylabel="Benefit"
      call xyplot$init(2,title,xlabel,ylabel,ll)
      call xyplot$reset_defaults(0,0,0,"secret",1," ",0,10.0," ",ll)
      call xyplot$build(j,x,z,01," ",1,ll)
      call xyplot$plot(ll)
2'6   continue
*
2(1   continue
      print ," o you want any graphs of cost of individual budget periods
                vs. individual task benefits?"
      read(5,20) answer
      if(answer .eq. "no") go to 301
      print ,"key in budget period desired."
      read(5,) i
      print ,"key in task benefit index."
      read(5,) n
*
*
      write(6,211) i,n
211   format(" ","To be plotted is a a graph of cost of year",i3,"
                     vs. benefit to task",i3," .")
*
      do 214 k=1,j
      z(k)=y(n,k)
      v(k)=yr(i,k)
214   continue
*
      write(6 ,30) (yr(i,l),l=1,j)
      write(6 ,31) (y(n,l),l=1,j)
      title =" "
      xlabel=" "
      cost="Cost of year"
      encode(xlabel,232) cost,i
232   format(a12,i2)
      ben="Benefit to task"
      encode(ylabel,233) ben,n
233   format(a15,i3)
      tle="COST OF YEAR"
      encode(til,234)tle,i
234   format(a12,i3)
      tse=" vs. BENEFIT TO TASK"
      encode(tpe,235) tse,n
235   format(a20,i3)
      encode(title,236) til,tpe
236   format(a15,a24)
      call xyplot$init(2,title,xlabel,ylabel,ll)
      call xyplot$reset_defaults(0,0,0,"secret",1," ",0,10.0," ",ll)
      call xyplot$build(j,v,z,01," ",1,ll)
      call xyplot$plot(ll)
275   continue
300   continue
          [illegible] 1
*
3 1   continue
*
      print ,"'o you want graphs showing cost of strategy for each budget period?"
      read(5,20) answer
      if (answer.eq. "no") go to 4(1
*
      zz=" "
      do 311 i=1,nper
      v(i)=float(nyr(i))
311   continue
      iz=0
      xlabel="Budget Period"
      ylabel="Cost"
      zz="COST vs. BUDGET PERIOD FOR STRATEGY"
      do 375 n=1,j
      do 320 i=1,nper
      z(i)=yr(i,n)
320   continue
      encode(title,325) zz,n
325   format(a35,i3)
*
      call xyplot$init(2,title,xlabel,ylabel,ll)
      call xyplot$reset_defaults(0,0,0,"secret",1," ",0,10.0," ",ll)
      call xyplot$build(nper,v,z,01," ",iz,ll)
      call xyplot$plot(ll)
      iz=iz+1
375   continue
*
401   continue
*
      call xyplot$done
      i=0
1000  continue
      close(9)
      return
      end
```

RESOURCE ALLOCATION METHODOLOGY FOR AIR FORCE R&D PLANNING

Volume 4: Computational Performance of RAM Programs

CONTENTS
Volume 4

I. INTRODUCTION

This volume is one of four that document ANSER's development of R&D resource allocation methodology (RAM) for the Director of Program Integration, AF/RDX. Volume 1 provides an overview of the work and its applications. Volume 2 describes the RAM technique and how to use the general-purpose computer programs that incorporate it. Volume 3 describes how to use the interactive computer program developed for use of the RAM within AF/RDX, and Volume 4 describes the way in which we tested its computational performance. Each volume emphasizes some particular aspect of our research and can be read independently of the others.

In this volume, we describe the results of our test program and document the way we conducted the tests. Chapter II contains our test plan, test results, and the inferences we can make from the results. In Chapter III, we document the procedures we used to make the tests. The results are readily understandable, although some knowledge of statistics would be helpful in understanding how we made inferences from the test data. The test procedures are of narrower interest and are documented primarily for our use in any further testing. They may be of some value, however, to others engaged in similar tests.

The primary reason for testing, other than to find any mistakes in the software, is to verify that the computer programs can handle the problems likely to be encountered. Capability is measured by determining the accuracy the program can achieve and the computer resources it requires as a function of problem size and complexity. We are satisfied that RAM/VM and RAM/GP (as defined in Volume 2 of this report) provide an excellent capability for routine use in the problems that are likely to occur in AF/RDX. Both programs have also performed well in other applications (see Volume 1). However, we still do not know all the limits of this software, since the cost of determining them would be prohibitive. Evaluation is difficult because we know of no other similar computer program that will accomplish the same tasks. Consequently, we can only compare the results of our work with the results of commercially available programs, which have less capability and must use simplified test problems. To achieve a higher level of confidence would require a substantially more expensive test program. We believe we have achieved an acceptable level of confidence with the testing done.

II. COMPUTATIONAL TEST RESULTS

A. Test Plan

After running the resource allocation algorithms on the Multics, we decided to formulate and carry out a test plan to evaluate the computer programs. The testing process had several goals. First, it was to verify the logic of the software. Second, the testing process was to define any limitations on the size or complexity of the problems that the programs could be expected to handle. Finally, it was to define conditions under which one programmed algorithm would be preferable to another.

The test plan used the Control Data Corporation's (CDC) APEX package,* a linear programming package that can solve integer programming problems. Unlike our resource allocation programs, the APEX cannot perform goal programming. However, because we knew of no other suitable alternative, we chose APEX as the standard against which the performance of our programs would be measured. We therefore did not use the goal programming capabilities of our programs.

The testing procedure consisted of running randomly generated problems in each of several problem sizes for each of our RAM programs+ and the APEX. The test problem size was the number of decision variables (alternatives to be considered in the resource allocation) for a problem. We devised random problems by using a random number generator to supply coefficients for problems with a specified number of decision variables. Then for each trial problem, we compared

---

* Hereafter, we refer to the package simply as APEX.

+ We also used our MIP2 program (see section II.B), but since we dropped the program partway through the testing it is not considered a significant part of the procedure.

and evaluated outputs for each of the RAM codes with those for APEX. We drew our conclusions from a statistical analysis of the results as well as from objective qualitative analysis. Finally, we tried to find standard test problems used by the operations research community against which to test the RAM codes, but we were not successful.

We encountered two principal difficulties during the testing process. First the use of APEX constrained the testing because it could not handle, at an affordable cost, problems of the size needed in our test procedure. Instead of comparing the results of our programs with APEX results for problems involving 100 to 300 decision variables, we had to limit comparisons to problems with less than 50 variables. Even for the smaller problems, APEX could not guarantee an optimal solution, but rather a solution within several percent of optimum. The second difficulty was the lack of a standard against which to measure goal programming. We are not likely to resolve this problem without development of additional independent goal programming algorithms.

B. Test Results

Three RAM programs were evaluated in the tests: RAM/VM, RAM/GP, and MIP2. MIP2 was our early experimental version of the resource allocation methodology, which was based on a variant of a simplex linear programming algorithm. Although this model will not be used in the future, partially as a result of these tests, the results are included here for comparison. RAM/VM and RAM/GP combine elements of goal programming and linear 0-1 programming and use the direct solution technique described in Volume 2. RAM/VM seeks the resource allocation that maximizes the overall benefit received subject to resource constraints. RAM/GP seeks the allocation that minimizes deviations from a set of specified, quantitative objectives. Both use a priority system for optimizing classes of objectives.

Table 1 shows the results of the trials of the RAM/GP, RAM/VM, MIP2, and APEX programs. For each trial, the same randomly generated problem was run on each of the four programs. On three occasions, the APEX did not obtain a solution within the specified time.

The column labeled "Percentage of Optimum (APEX)" indicates within what percentage of optimum the APEX solution was guaranteed. In Trial 5, for example, the best APEX solution found is guaranteed to be within 5.89 percent of the value of the objective function at the true optimal solution. As a consequence of this, the product of 1.0589 and the objective function value (1,363.428), or 1,443,734, is an upper bound for the value of the objective function evaluated at the actual optimal solution.

Note that the MIP2 program rarely reached a solution, returning instead a message that the program had been unable to find a feasible solution. This inability to find feasible solutions is caused by the inability of the program to maintain required solution criteria for each interaction. It does not indicate that the problem had no feasible solution. We had suspected this problem before testing but had not realized its extent until partway through the testing procedures. When the MIP2 program did find a solution, it matched those found by the other programs. However, we decided that the extent of MIP2's inability to find feasible solutions warranted dropping it from the testing plan.

We applied the Wilcoxon Matched-Pairs, Signed-Rank test to the data in Table 1. We used this nonparametric statistical test to compare the results that RAM/GP and RAM/VM provide with those provided by APEX. The advantage of the nonparametric test for paired differences is that it does not require the assumption

TABLE 1
SUMMARY OF TRIAL RESULTS

Trial Number	Number of Variables	Value of Objective Function at Best Solution Obtained				Percentage of Optimum (APEX)
		RAM/GP	RAM/VM	MIP2	APEX	
1	12	630.952		--	630.952	1.14
2	15	631.946	Not Run	--	604.211	9.50
3	18	1,461.984		1,492.763	1,492.703	Optimal
4	21	1,689.535	1,689.535	1,689.520	1,689.535	Optimal
5	21	1,380.276	1,380.276	NFS*	1,363.428	5.89
6	21	1,265.859	1,265.859	NFS	1,265.859	5.97
7	21	1,453.479	1,453.479	NFS	1,417.578	6.72
8	21	1,354.015	1,354.015	NFS	1,232.959	9.49
9	21	1,467.194	1,467.194	NFS	1,467.194	1.25
10	21	1,513.992	1,513.992	NFS	1,513.992	
11	21	1,185.199	1,185.199	NFS	1,185.199	1.40
12	21	1,361.286	1,361.286	NFS	1,361.286	3.05
13	21	1,073.619	1,073.619	NFS	1,047.597	7.83
14	45	2,510.933	2,510.933	NFS	2,393.704	4.18
15	45	2,735.956	2,735.956	NFS	2,735.054	3.14
16	45	2,854.954	2,854.954	--	2,869.605	3.54
17	45	2,584.174	2,584.174	--	2,584.174	1.22
18	45	3,095.392	3,095.392	--	3,095.392	1.19
19	45	2,339.578	2,339.578	--		
20	45	2,220.971	2,220.971	--	2,179.195	3.71
21	45	2,500.680	2,500.680	--		
22	45	2,499.623	2,499.623	--	2,622.231	0.42

*No feasible solution.

that the underlying population of differences is normally distributed. The nonparametric test makes no assumption about the population distribution. We could not divide the trials into groups according to the number of variables because the resulting data sets would not be large enough to be statistically meaningful.

Table 2 shows the data for applying the Wilcoxon Matched-Pairs, Signed Rank Test to the values of the objective functions provided by RAM/GP and RAM/VM. The test is carried out in terms of the paired differences, $d = x_1 - x_2$, where the d values represent differences between two observations on the same individual or object. In this case, the difference is between the results obtained by two computer programs given the same randomly generated problem. The absolute values of the differences are ranked from 1 to n, with the smallest difference being assigned the rank of 1. Each rank is then given the sign (either + or -) of the associated value of d. If there are ties in ranking, the mean rank value is assigned to the tied items. If, for example, the sixth and seventh ranked items are tied, a rank of $(6 + 7)/2 = 6.5$ is assigned to each. If the difference between paired observations is 0, that item is dropped and the number of differences reduced by one. Since Table 2, has 10 zeros in the difference column, the effective sample size is $20 - 10 = 10$.

As indicated in the last two columns of Table 2, the sums of the ranks are obtained separately for the positive and negative differences. These sums, Σ ranks (+) and Σ ranks (-), form the basis for the null hypothesis H_0: Σ ranks (+) = Σ rank (-). Specifically, the null hypothesis is that the positive and negative differences in the population are symmetrically distributed about a mean of 0. The smaller of the two ranked sums is called Wilcoxon's T statistic and is the test statistic. Hence, in Table 2, the test statistic is T = Σ ranks (-) = 16.

TABLE 2
DATA FOR THE WILCOXON MATCHED-PAIRS, SIGNED-RANK TEST

Trial Number	RAM/GP Solution X_1	APEX Solution X_2	Difference $d = X_1 - X_2$	Rank of d	Signed Rank	
					Rank (+)	Rank (–)
1	630.952	630.952	0			
2	631.946	604.211	+27.735	4	4	
3	1,461.984	1,492.763	–30.779	5		5
4	1,689.535	1,689.535	0			
5	1,380.276	1,363.428	+16.848	2	2	
6	1,265.859	1,265.859	0			
7	1,453.479	1,417.578	+35.901	6	6	
8	1,354.015	1,232.959	+121.056	9	9	
9	1,467.194	1,467.194	0			
10	1,513.992	1,513.992	0			
11	1,185.199	1,185.199	0			
12	1,361.286	1,361.286	0			
13	1,073.619	1,047.597	+26.022	3	3	
14	2,510.933	2,393.704	+117.229	8	8	
15	2,735.956	2,735.956	0			
16	2,854.954	2,869.605	–14.651	1		1
17	2,584.174	2,584.174	0			
18	3,095.392	3,095.392	0			
20	2,220.971	2,179.195	+41.776	7	7	
22	2,499.623	2,622.231	–122.608	10		10
					39	16

Table 3 gives the critical values of T. For sample size N=10 at the 0.10 level of significance, T can be no greater than 10. Note that Table 3 presents the maximum values T can have and still be considered significant at the stated significance level. Thus, since the calculated value of T(16) exceeds 10, we cannot reject the null hypothesis of identical population distributions. Hence, we concluded that the performance of RAM/GP and RAM/VM did not differ significantly in the values obtained for the objective functions for the trial problems. We reached the same conclusion when comparing RAM/VM with APEX and RAM/GP with APEX.

The value of these results is obscured by the lack of certainty as to whether APEX was achieving optimal solutions for the trial problems. That APEX guarantees a solution to be within a specified percentage of optimum does not mean that the solution is not optimal. Nevertheless, except for Trials 3 and 4, we cannot prove that the solutions are optimal.

Table 4 presents upper bounds for the value of the objective function at optimal solutions and the ratio of the RAM/GP solution to the upper bound. For 21-variable problems, the mean of the ratios is 0.973 with a standard deviation of 0.22 and variance of 0.0004. For 45-variable problems, the mean of the ratios is 0.977 with a standard deviation of 0.019 and variance of 0.0003. The overall mean of the ratios is 0.975 with a standard deviation of 0.020 and variance of 0.0004. Using this information, we can establish, for the overall set of 22 samples, a confidence of 0.995 that 70 percent of the sample distribution of ratios are between the values of 0.944 (minimum value occurring) and 1.007 (maximum value occurring). We can establish a confidence of 0.95 that 80 percent of the

TABLE 3
CRITICAL VALUES OF T* IN THE
WILCOXON MATCHED-PAIRS, SIGNED-RANK TEST

N	Level of significance for one-tailed test			
	.05	.025	.01	.005
	Level of significance for two-tailed test			
	.10	.05	.02	.01
5	0	--	--	--
6	2	0	--	--
7	3	2	0	--
8	5	3	1	0
9	8	5	3	1
10	10	8	5	3
11	13	10	7	5
12	17	13	9	7
13	21	17	12	9
14	25	21	15	12
15	30	25	19	15
16	35	29	23	19
17	41	34	27	23
18	47	40	32	27
19	53	46	37	32
20	60	52	43	37
21	67	58	49	42
22	75	65	55	48
23	83	73	62	54
24	91	81	69	61
25	100	89	76	69
26	110	98	84	75
27	119	107	92	83
28	130	116	101	91
29	140	126	110	100
30	151	137	120	109
31	163	147	130	118
32	175	159	140	128
35	213	195	173	159
40	286	264	238	220
45	371	343	312	291
50	466	434	397	373

*The symbol T denotes the smaller sum of ranks associated with differences that are all of the same sign. For any given N (number of ranked differences), the obtained T is significant at a given level if it is equal to or less than the value shown in the table.

TABLE 4
STATISTICS ON SOLUTION RATIOS

Trial Number*	Number of Variables	RAM/GP Solution	RAM/VM Solution	APEX Solution	Percentage of Optimum (APEX)	Upper Bound†	RAM/GP Upper Bound†	Statistics for Ratio
1	12	630.952	--	630.952	1.14	638.145	0.989	
2	15	631.946	--	604.211	9.50	661.611	0.955	
3	18	1,461.984	--	1,492.763	Optimal	1,492.763	1.000	
4	21	1,689.535	1,689.535	1,689.535	Optimal	1,689.535	1.000	
5	21	1,380.276	1,380.276	1,363.428	5.89	1,443.734	0.956	
6	21	1,265.859	1,265.859	1,265.859	5.97	1,341.431	0.944	M = 0.9730
7	21	1,453.479	1,453.479	1,417.578	6.72	1,512.839	0.961	J = 0.0220
8	21	1,354.015	1,354.015	1,232.959	9.49	1,349.967	1.003	J^2 = 0.0004
9	21	1,467.194	1,467.194	1,467.194	1.25	1,485.534	0.988	
10	21	1,513.992	1,513.992	1,513.992	3.56	1,567.890	0.966	
11	21	1,185.199	1,185.199	1,185.199	1.40	1,201.792	0.986	
12	21	1,361.286	1,361.286	1,361.286	3.05	1,402.805	0.970	
13	21	1,073.619	1,073.619	1,047.597	7.83	1,129.624	0.950	
14	45	2,510.933	2,510.933	2,393.704	4.18	2,493.671	1.007	
15	45	2,735.956	2,735.956	2,735.054	3.14	2,820.935	0.970	M = 0.9770
16	45	2,854.954	2,854.954	2,869.605	3.54	2,971.189	0.961	J = 0.0190
17	45	2,584.174	2,584.174	2,584.174	1.22	2,615.701	0.988	J^2 = 0.0003
18	45	3,095.392	3,095.392	3,095.392	1.19	3,154.204	0.981	
20	45	2,220.971	2,220.971	2,179.195	3.71	2,260.043	0.983	
22	45	2,499.623	2,499.623	2,622.231	0.42	2,633.244	0.949	
								M = 0.9750
								J = 0.0200
								J^2 = 0.0040

*Trials 19 and 21 are omitted because APEX did not reach a solution.

†Computed by multiplying: $\frac{1.0 + \text{percentage of optimum (APEX)}}{100}$ x APEX solution.

Thus, for trial 22, 2,633.244 = (1.0042) x (2,622.231).

The occurrence of meaningful values greater than 1.0 among entries in this column is, by definition of upper bound, *impossible. Nevertheless, they occur.* We do not know whether they are a result of roundoff error, errors in the returned value of percentage of optimum (APEX), or of other causes. Since we did not know the cause of entries greater than 1.0, we did not tamper with data by resetting the value of such entries to 1.0.

distribution of ratios are greater than 0.944. These values are taken from standard tables of nonparametric tolerance limits.[1]

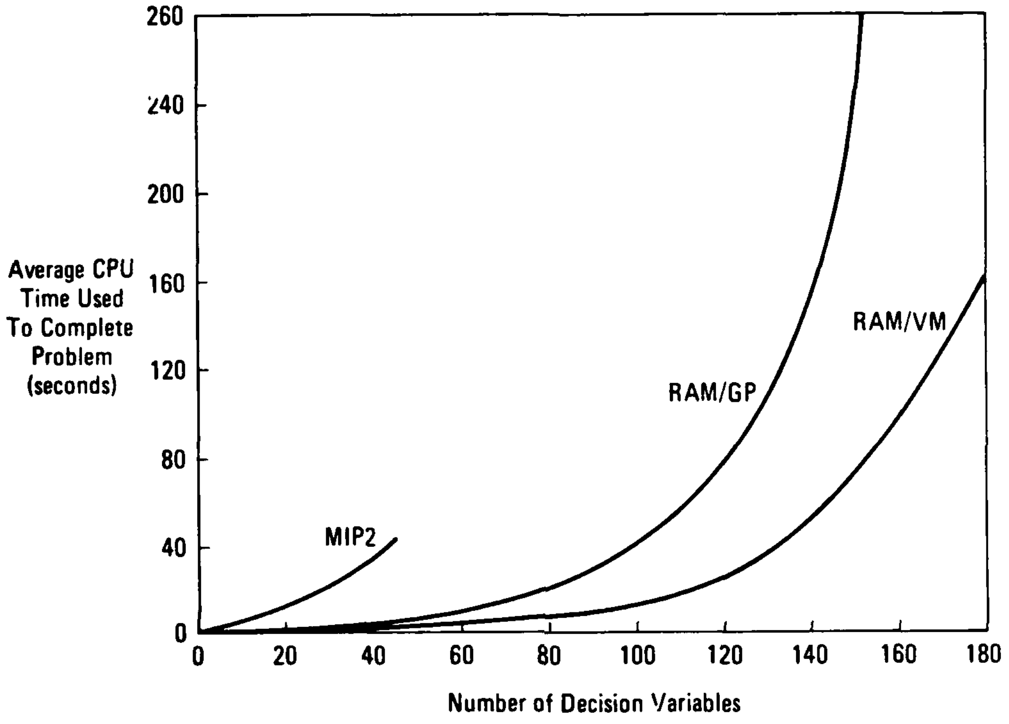
These results indicate that a user of RAM/GP or RAM/VM, can have a high degree of confidence that results from these programs are reasonably close to optimum for problems involving no more than 45 decision variables. However, we have no basis for extrapolating these results to larger problems.

Figure 1 shows the average time used by RAM/GP, RAM/VM, and MIP2 to complete a problem as a function of the number of decision variables within the problem. The figure implies that the RAM/GP program has a reasonable running time for problem sizes of 200 to 250 decision variables. For RAM/VM, the running time would be considerably less for problems of the same size. The sketchy data for MIP2 seem to indicate that this program would be too slow to run very large problems.

The conclusions that we can draw from the test results are reasonably clear. The RAM/GP and RAM/VM programs may be used with confidence in problems that have up to 45 decision variables and that do not involve any goal programming. The results obtained may or may not be optimal, but they will in all likelihood be quite close to optimal. The testing procedure could not evaluate the optimization capability of RAM/GP and RAM/VM for problems larger than 45 decision variables or for those involving goal programming. The MIP2 program was found to be incomplete and unsuitable for use.

[1] R. E. Walpole and R. H. Myers, *Probability and Statistics for Engineers and Scientists*, 2nd Ed., New York: MacMillan Publishing Co., Inc., 1978.

FIGURE 1
TIME USED TO COMPLETE
PROBLEM AS A FUNCTION OF PROBLEM SIZE



Unfortunately, we could not meet all the test program objectives. However, even the limited results obtained are useful in developing some confidence in the programs and reasonable expectations for their performance.

III. TEST PROGRAM OPERATION

A. Test Problem Generation

The problems generated for the test program took the following form:

$$\text{Minimize:}\quad Z = \sum_{j=1}^{NG} w_j n_j$$

Subject to:

$$\sum_{i=1}^{NVAR} C_{im} X_i \leq B_m \qquad m = 1,NPER$$

$$\sum_{i=1}^{NVAR} a_{ij} X_i + n_j \geq A_j \qquad j = 1,NG$$

$$\sum_{X_i \varepsilon GR_x} X_i \leq 1 \qquad k = 1,NGR$$

$$X_i \ \varepsilon\ \{0,1\}\ \forall i \qquad i = 1,NVAR$$

$$n_j \geq o\ \forall_j \qquad j = 1,NG$$

An input generator was developed that created a set of pseudorandom numbers and formatted them to be used as input data by RAM/VM and RAM/GP. The input generator created a "problem" based on a set of given parameters and a "seed." The seed,* which was given a unique value for each problem, was the basis for the random number generator. The parameters for the input generator were the number of (a) 0-1 variables (NVAR), (b) goals (NG), (c) mutually exclusive groups (NGR), (d) priority levels (NP), and (e) time periods and values for B_m (m=1,NPER). The values for A_j (j=1,NG) were based on the value of the parameter, NGR (for most test problems, A_j=119NGR). The random number generator provided the values for the coefficients C_{im} and a_{ij}.

*For an explanation of the use of "seed" numbers in generating random numbers, see Thesen, Arne, *Computer Methods in Operations Research*, New York: Academic Press, 1978.

PRECEDING PAGE BLANK-NOT FILMED

All test problems had four goals (NG=4), one priority level (NP=1), four times periods (NPER=4), and three 0-1 variables in each mutually exclusive group (therefore, NVAR=3*NGR). NVAR and NGR were varied during the test program that determined the size of the problem. Unique sets of coefficients (C_{im} and A_{ij}) were generated for each problem. Within each problem size, the values for B_m were also varied. Table 5 shows a sample problem created by the input generator.

TABLE 5
SAMPLE DATA CREATED BY INPUT GENERATOR

```
     6       4       1       4       2       1
     1       1       1       1       1       1       1      1
  92.480  20.398   9.036  64.670  32.761  35.364
  57.644  21.867  49.860  79.290  90.894  92.023
  79.913  90.833  27.228  44.428   2.967  36.395
   7.520  38.274  44.577  17.664  56.538  88.691
     1       1       1       2       2       2
     1       2       3       1       2       3
  30.000  45.000  35.000  40.000
  42.017  28.396  13.482   8.219  92.480  20.398
   9.036  64.670  32.761  35.364  57.644  21.867
  49.860  79.290  90.894  92.023  79.913  90.833
  27.228  44.428   2.967  36.395   7.520  38.274
 238.000 238.000 238.000 238.000
```

Because the input generator created the problems in a format compatible with RAM/VM and RAM/GP, we had to develop preprocessors (reformatting programs) to input the same problems to MIP2 and APEX. These preprocessors read in the data as created by the input generator and reformatted it to be compatible with MIP2 and APEX. Table 6 shows the data provided in Table 5 as reformatted for APEX.

Listings of the computer codes that generated the data and reformatted them are available from ANSER. The codes used were implemented on the CDC system. Minor changes have been made in the coding to make it compatible with the CDC system and to ensure that the problems generated for APEX were the same as those generated on the Multics system for RAM/VM and RAM/GP.

B. Test Problem Solution

APEX is a commercially available optimization program developed by CDC for use on their operating systems. During the test program, we used the APEX Mixed Integer Programming option. This option uses a branch-and-bound algorithm to solve the mixed integer problem.

1. Run Specifications for the Test Program

We ran APEX as a batch job and submitted the problems using either a "single solve card" or a "control program." Examples of the input files for each are shown in Tables 7 and 8.

---

*All development and applications of RAM programs were done on the Multics computer system as installed on the Air Force's Honeywell Series 68/Level 60 computer.

TABLE 6
DATA FORMATTED FOR APEX USE

```
NAME           MOFA
ROWS
 N  OBJFUNC
 L     1CROW
 L     2CROW
 L     3CROW
 L     4CROW
 E     1SOS
 E     2SOS
 G     1AROW
 G     2AROW
 G     3AROW
 G     4AROW

COLUMNS
       1COL       1CROW       92.480
       1COL       2CROW       57.644
       1COL       3CROW       79.913
       1COL       4CROW        7.520
       1COL       1AROW       42.017
       1COL       2AROW        9.036
       1COL       3AROW       49.860
       1COL       4AROW       27.228
       1COL       1SOS         1.000
       2COL       1CROW       20.398
       2COL       2CROW       21.867
       2COL       3CROW       90.833
       2COL       4CROW       38.274
       2COL       1AROW       28.396
       2COL       2AROW       64.670
       2COL       3AROW       79.290
       2COL       4AROW       44.428
       2COL       1SOS         1.000
       3COL       1CROW        9.036
       3COL       2CROW       49.860
       3COL       3CROW       27.228
       3COL       4CROW       44.577
       3COL       1AROW       13.482
       3COL       2AROW       32.761
       3COL       3AROW       90.894
       3COL       4AROW        2.967
       3COL       1SOS         1.000
       1SC        1SOS         1.000
       4COL       1CROW       64.670
       4COL       2CROW       79.290
       4COL       3CROW       44.428
       4COL       4CROW       17.664
       4COL       1AROW        8.219
       4COL       2AROW       35.364
       4COL       3AROW       92.023
       4COL       4AROW       36.395
       4COL       2SOS         1.000
       5COL       1CROW       32.761
       5COL       2CROW       90.894
       5COL       3CROW        2.967
       5COL       4CROW       56.538
       5COL       1AROW       92.480
       5COL       2AROW       57.644
       5COL       3AROW       79.913
       5COL       4AROW        7.520
       5COL       2SOS         1.000
       6COL       1CROW       35.364
       6COL       2CROW       92.023
       6COL       3CROW       36.395
       6COL       4CROW       88.691
       6COL       1AROW       20.398
       6COL       2AROW       21.867
       6COL       3AROW       90.833
       6COL       4AROW       38.274
       6COL       2SOS         1.000
       2SC        2SOS         1.000
       1UDV       1AROW        1.000            OBJFUNC         1.000
       2UDV       2AROW        1.000            OBJFUNC         1.000
       3UDV       3AROW        1.000            OBJFUNC         1.000
       4UDV       4AROW        1.000            OBJFUNC         1.000
RHS
    RHS           1CROW       30.000
    RHS           2CROW       45.000
    RHS           3CROW       35.000
    RHS           4CROW       40.000
    RHS           1SOS         1.000
    RHS           2SOS         1.000
    RHS           1AROW      238.000
    RHS           2AROW      238.000
    RHS           3AROW      238.000
    RHS           4AROW      238.000
BOUNDS
 SI LIMITS        1COL                          1SC
 SI LIMITS        4COL                          2SC
ENDATA
```

TABLE 7
SAMPLE INPUT FILE FOR RUNNING APEX
USING "SINGLE SOLVE CARD"

```
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible] 170 [illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible] 260 /[illegible]
[illegible] 270 /[illegible]
```

TABLE 8
SAMPLE INPUT FILE FOR RUNNING APEX
USING "CONTROL PROGRAM"

```
00100 /JOB
00110 RCH1,T50,P4.
00120 USER(S0106VE,AMSEDCC,ME)
00130 ATTACH(APEX/UN=LIBRARY)
00140 GET,TAPE1=TAPE6.
00145 DEFINE,TEMP.
00150 RFL,100000.
00160 REDUCE,-.
00170 APEX(C,SOF=TEMP)
00210 DAYFILE,DAY.
00220 REPLACE,DAY.
00230 EXIT.
00240 DAYFILE,DAY.
00250 REPLACE,DAY.
00260 /EOR
          INPUT       $
          SET         RNDIR       MIN
          SET         RNOBJ       $
          SET         RNRMS       $
          SET         RNEND       $
          SELECT
          SET         RPOBEND     -4640.
          CRASH
          MIXINT
          OUTPUT      FULL        TEMP
          EXIT
00265 /EOR
00270 /EOF
```

APEX can be submitted using the "single solve card" for most problems. The solve card allows the user to set the direction of optimization (MIN or MAX) and select any of a number of options. Line 00170 of Table 7 is an example of the "single solve card." "MIN" informs APEX that the problem being submitted is a minimization problem. "MIP" directs APEX to use the mixed integer programming option. The "SOF" option directs APEX to write all output into a direct access file, defined here as TEMP. The "RL" option establishes a resource limit that, if exceeded, would cause APEX to discontinue processing and exit. Two additional options that should be mentioned are "OPT" and "SV." "OPT" directs APEX, when using the MIP option, to find and prove the optimal solution. If the "OPT" option is not included as an argument on the solve card, APEX will terminate the search for an optimal solution after finding a solution within 10 percent of optimum. "SV" will direct APEX to save all the data required to restart the problem where it was terminated. Other available options are described in the *APEX Reference Manual* furnished by CDC.

Use of an APEX "control program" allows greater flexibility and control of the APEX solution system. The example in Table 8 shows one way in which it was used during the test program. "RPOBBND" is the lower bound on the objective function and places a bound on many of the branches APEX must explore in solving the mixed integer problem. The solution to the problem, as determined by RAM/VM and RAM/GP, was input in the "control program" as "RPOBBND" to limit the number of branches APEX needed to consider. For minimization problems, the value of "RPOBBND" is complemented (-). The "control program" also selects the mixed integer option (MIXINT) and directs APEX to write the output into a file, defined here as TEMP.

2. Representation of Decision Variables

We used two approaches in setting up the problem in the proper format for APEX to handle the constraints:

$$\sum_{X_i \in GR_k} X_i \leq 1 \qquad k=1,NGR$$

and $$X_i \in \{0,1\} \; \forall i \qquad i=1,NVAR$$

One approach was to declare the 0-1 variables as bivalent variables; the other was to use special ordered sets. When using bivalent variables, all 0-1 variables must be declared in the bounds section of the input data. The equations establishing the mutually exclusive group must be of the less-than-or-equal-to type, as declared in the rows section. To use special ordered sets, an additional variable must be added to each mutually exclusive group. (These added variables are labeled 1SC and 2SC in Table 6.) The constraint then becomes an equal-to type, and APEX must select exactly one variable from each mutually exclusive group. The appearance of the added variable in the solution means that its associated mutually exclusive group is not represented in the solution. Although use of the special ordered sets requires that additional variables be included in the problems, we found that APEX generally solves the problems more efficiently when they are structured as special ordered sets than when they are set up with bivalent variables. Therefore, most problems solved during the test program used the special ordered sets.

PRIMARY DISTRIBUTION LIST FOR TDN 80-1

ORGANIZATION	NUMBER OF COPIES
OSD/AE (Lt Col J. Gross)	1
AF/RDX (Brig Gen M. Roger Peterson)	1
(Lt Col F. Gerken)	3
AF/SAMI	1
OSD/PA&E (Capt B. Berkowitz)	1
AF/RDP (Mr. G. Fisher)	1
ASD/ENASC (Mr. Larry Beasley)	1
AF/XOXIM (Maj E. Wilkins)	1
AFSC/XRS (Lt Col G. Hollobaugh	1
SD/YLXA (Lt Michele Focht)	1
DTIC	2
ANSER	
Library	5
Reserve Stock	20
Master Copy (1)	

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER TDN 80-1	2. GOVT ACCESSION NO. AD-A096 546	3. RECIPIENT'S CATALOG NUMBER --
4. TITLE (and Subtitle) Resource Allocation Methodology for Air Force R&D Planning		5. TYPE OF REPORT & PERIOD COVERED Division Note
		6. PERFORMING ORG. REPORT NUMBER TDN 80-1
7. AUTHOR(s) G. Cooper, S. Adams, J. Clary, J. Perlis		8. CONTRACT OR GRANT NUMBER(s) F-49620-77-C-0025
9. PERFORMING ORGANIZATION NAME AND ADDRESS ANSER (Analytic Services Inc.) 400 Army Navy Drive Arlington, Virginia 22202		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Directorate of Program Integration Headquarters United States Air Force Washington, D.C. 20330		12. REPORT DATE June 1980
		13. NUMBER OF PAGES 130
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

rescurce allocation, heuristics, goal programming, vector maximization, integer programming, R&D project selection, budgeting

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

~~During fiscal 1979~~, the Director of Program Integration, AF/RDX, tasked ANSER to identify and develop a methodology for allocation of funds among Air Force research, development, and acquisition programs. This report, which consists of four volumes, describes work accomplished in response to that tasking and discusses the resource allocation methodology that resulted. Volume I is an overview of the work. Volume 2 describes the resource allocation techniques and the general-purpose computer programs that incorporate it. Volume 3 discusses how to use the interactive computer program developed for

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE (over)

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

use of RAM within AF/RDX. Volume 4 reports on tests made to determine the computational performance of the methodology.

DATE
ILMED
-8